2011-06

Agent-based simulation and analysis of a defensive UAV swarm against an enemy UAV swarm

Munoz, Mauricio F.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

AGENT-BASED SIMULATION AND ANALYSIS OF A DEFENSIVE UAV SWARM AGAINST AN ENEMY UAV SWARM

by

Mauricio F. Munoz

June 2011

Thesis Advisor:	Timothy H. Chung
Second Reader:	Michael P. Atkinson

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
8–6–2011	Master's Thesis	6-21-2009—6-17-2011

4. TITLE AND SUBTITLE

Agent-Based Simulation and Analysis of a Defensive UAV Swarm Against an Enemy UAV Swarm

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Mauricio F. Munoz

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number_N/A_.

14. ABSTRACT

Unmanned systems, including unmanned combat aerial vehicles (UCAVs), are increasingly important in military operations. Given the growth of unmanned systems technology worldwide, these systems may increasingly pose a real threat to U.S. and Allied forces in the near future. This thesis proposes a future concept of employing a defensive UCAV swarm, launched from a friendly sea-based platform. To simulate this defensive swarm system, an agent-based simulation model was developed, and appropriate designs of experiments and statistical analyses were conducted. The investigated factors were drawn from the literature review to create several experimental designs with the objective of identifying the significant design factors of the Blue UCAV system. The result of the analysis shows that only five of the eleven candidate factors analyzed are significant, which can be used to inform the engineering specification of preliminary requirements for potential future development.

15. SUBJECT TERMS

UAV, UCAV, UAV Swarm, Agent-based simulation, Repast Symphony

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	109	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

AGENT-BASED SIMULATION AND ANALYSIS OF A DEFENSIVE UAV SWARM AGAINST AN ENEMY UAV SWARM

Mauricio F. Munoz
Lieutenant, Chilean Navy
Naval Electronic Engineer

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2011

Author: Mauricio F. Munoz

Approved by: Timothy H. Chung
Thesis Advisor

Michael P. Atkinson
Second Reader

Robert Dell
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Unmanned systems, including unmanned combat aerial vehicles (UCAVs), are increasingly important in military operations. Given the growth of unmanned systems technology worldwide, these systems may increasingly pose a real threat to U.S. and Allied forces in the near future. This thesis proposes a future concept of employing a defensive UCAV swarm, launched from a friendly sea-based platform. To simulate this defensive swarm system, an agent-based simulation model was developed, and appropriate designs of experiments and statistical analyses were conducted. The investigated factors were drawn from the literature review to create several experimental designs with the objective of identifying the significant design factors of the Blue UCAV system. The result of the analysis shows that only five of the eleven candidate factors analyzed are significant, which can be used to inform the engineering specification of preliminary requirements for potential future development.

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

Unmanned systems, including unmanned combat aerial vehicles (UCAVs), are increasingly important in military operations given their versatility in mission capabilities and their ability to reduce risk to manned forces. However, given the growth of unmanned systems technology worldwide, these systems may increasingly pose a real threat to U.S. and allied forces in the near future. This thesis analyzes an example of such a threat: a coordinated attack by a large number of enemy UCAVs against a surface combatant. An example of a UCAV is the IAI *Harop*, which is an expendable UCAV designed to detect and destroy radar emissions.

This thesis proposes a future concept of employing a defensive UCAV swarm, launched from a friendly sea-based platform, as a possible solution to this emerging threat. To simulate this defensive swarm system, an agent-based simulation model is developed, and appropriate designs of experiments and statistical analyses are conducted. Realistic data used to build the agent behaviors and characteristics in the simulation are obtained from various open sources.

The investigated factors are drawn from the literature review to create several experimental designs with the objective of identifying the significant design factors of the Blue UCAV system. The desired outcome is the construction of a prediction model to predict the probability of killing an enemy UCAV, thereby providing a measure of effectiveness against the enemy UCAV swarm.

The result of the analysis shows that only five of the eleven candidate factors analyzed are significant. Of the five significant factors, four correspond to specific controllable design factors of the UCAV defense system which can be used to inform the engineering specification of preliminary requirements for potential future development.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction **1**
1.1 Background 1
1.2 Objectives 2
1.3 Literature Review 2
1.4 Scope, Limitations and Assumptions 5
1.5 Contribution of this thesis 6
1.6 Organization of the thesis 7

2 Model Formulation **9**
2.1 Real World Scenario 9
2.2 Agent Descriptions 10
2.3 Simulation Model 13
2.4 Development of the Model 19
2.5 Different Modeling Approaches 19

3 Experimental Design **25**
3.1 List of Factors 25
3.2 Design of Experiments 33

4 Analysis **37**
4.1 Objective of the Analysis 37
4.2 *I*-optimal Design 42
4.3 Space-filling design 48

5 Conclusions and Recommendations **55**
5.1 Conclusions 55
5.2 Recommendations 57
5.3 Follow up Studies 58

List of References **61**

Appendices **65**

A Design Matrices **65**
A.1 Screening Design Tables 65
A.2 *I*-optimal Design Tables. 65
A.3 Sphere Packing Design Table 65
A.4 Latin Hypercube Design Table 65

B Model Source Code **75**
B.1 First Context Creator 75
B.2 Blue HVU Class 77
B.3 Blue UCAV Class 81
B.4 Red UCAV Class 85

Initial Distribution List **91**

List of Figures

Figure 1.1 A graphical representation of the course of study for this thesis 7

Figure 1.2 Screen shot of the implemented agent-based simulation model 8

Figure 2.1 Scenario Set Up 10

Figure 2.2 Diagram of the process to get the final model 11

Figure 2.3 Blue HVU behavior diagram 12

Figure 2.4 Blue UCAV behavior diagram 13

Figure 2.5 Angle between Red UCAV and HVU 14

Figure 2.6 Red UCAV behavior diagram 14

Figure 2.7 Axis coordinate orientation 15

Figure 2.8 Model Development 19

Figure 2.9 First Analysis Approach 23

Figure 2.10 Results using first Approach 23

Figure 3.1 Speeds for existing UAV and UCAV systems 26

Figure 3.2 Representative anti-air missile blast range and explosive material . . . 27

Figure 3.3 Endurance times for various UAVs or UCAVs systems 28

Figure 4.1 Snap shot of the Fit a Model Screen 38

Figure 4.2 Snap shot of *Stepwise* tool window 39
Figure 4.3 Normality Analysis 41
Figure 4.4 Residual by Predicted Plot 41
Figure 4.5 Durbin-Watson Statistic 42
Figure 4.6 Summary of Fit table 43
Figure 4.7 Parameters Estimates Table 44
Figure 4.8 Summary of Fit Table for the *I*-optimal design 45
Figure 4.9 Normality assumption fail in *I*-optimal design 45
Figure 4.10 Fitted values vs actual values *I*-optimal design 47
Figure 4.11 Fitted values vs actual values Latin Hypercube test 48
Figure 4.12 Profiler plots from the *I*-optimal design 49
Figure 4.13 Correlation Matrix of the Compiled design 50
Figure 4.14 Comparison of two prediction Models 51
Figure 4.15 Prediction Profiler of the prediction model 53

Figure 5.1 Diagram of proposed future study 59

List of Tables

Table 3.1 Summary Table of Experimental Factors and Levels 31

Table 4.1 Latin Hypercube table 48

Table 4.2 Values to obtain a predicted probability of 99.9% 52

Table 4.3 Values to obtain a predicted probability of 90% 52

Table 4.4 Values to obtain a predicted probability of 80% 52

Table 4.5 Values to obtain a predicted probability of 50% 52

Table A.1 Screening-1 66

Table A.2 Screening-2 67

Table A.3 Screening-3 68

Table A.4 Screening-4 69

Table A.5 Screening-5 70

Table A.6 *I*-optimal-1 71

Table A.7 *I*-optimal-2 72

Table A.8 *Sphere Packing* 73

Table A.9 *Latin Hypercube* 74

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

It would not be fair unless I mentioned all my professors who shared their knowledge over these two years. It is because of them I could finish this thesis in the best way. Some of them need special recognition, especially my thesis advisor Tim Chung, who from the first moment gave me 110% support and confidence to make this process an amazing experience and who gave me the tools necessary to finish this work. Also, to my editor, Barbara Young, who was also my English professor when I arrived at NPS. Without her, probably this work would be unreadable. I also would like thank my second reader, Mike Atkinson, for his advice and because he was my first instructor in JAVA, the computer language that I used for my simulation model. And finally, and not for that less important since actually she is the most important person in this process, my wife Mariella. Without her constant support, understanding and love this work would have been impossible to finish. And, mostly because she gave me our most valuable treasure, our son Mauricio Jr.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Background

Unmanned systems, including unmanned combat aerial vehicles (UCAVs), are increasingly important in military operations, given their versatility in mission capabilities and their ability to reduce risk to manned forces. Given the growth of technology worldwide, current and future adversaries may use these unmanned systems against the U.S. and/or allied nations in armed conflict. Furthermore, the asymmetric nature of modern warfare is exposing the effectiveness of high numbers and low cost UCAVs against deployed defenses. Therefore, it is necessary to develop a viable defensive system against this emerging threat. This thesis focuses on a defense system to be used against a UCAV swarm, which can be described as a large number of enemies or "Red" UCAVs programmed to engage a certain target, in this case a high value unit (HVU) deployed in open waters. The idea of a Red UCAV swarm is to saturate the anti-air (AA) defense systems of the HVU by a sustained attack. An example of such a UCAV is the IAI *Harop*, developed by Israel Aerospace Industries [1]. The IAI *Harop* is an expendable UCAV designed to detect and destroy radar emissions. This UCAV is also equipped with a 70-pound high-explosive warhead to damage the target. According to open sources [2], this system has been sold to countries like China, South Korea, Turkey, India and Chile[1]. Among possible solutions to a Red UCAV threat, one future concept is to have a *defensive* UCAV swarm to counter the saturation attack. A potential deployment approach requires that this defensive or "Blue" UCAV swarm system be sea based, which means that it needs to be launched from an HVU. Other possible solutions include improved capabilities of existing systems or future measures such as directed energy weapons. Investigation of such alternatives are beyond the scope of this thesis.

In this thesis, the proposed approach of employing a defensive swarm of UAVs to counter the Red UCAV swarm threat is investigated using methods of agent-based simulation and design of experiments. The scenario is implemented in a freely available, open source simulation environment called Repast Simphony [3] which allows the user to define unique and customized behavior for each agent in the Java programming language.

[1]Chilean involvement with the IAI *Harop* is limited to system evaluation only.

1.2 Objectives

In this thesis, the author seeks to analyze the Blue UCAV defense system, identify the significant design factors, and determine the most effective ways to use them. In order to do this, the following objectives are proposed:

- Define Measure of Effectiveness (MOEs)
- Identify the behavior for each agent in the simulation
- Build an agent-based simulation in Repast Simphony
- Identify important design factors
- Identify uncontrollable factors
- Analyze the data from the simulation

To achieve these objectives, a thorough literature review of available open and unclassified sources is conducted and presented in the following section to understand the nature of swarm UCAVs. Other material concerning unmanned aerial systems is examined as well.

1.3 Literature Review

Numerous research efforts on unmanned aerial vehicles (UAVs) have been conducted over the years. For example, Berner [4] analyzed the effective use of multiple UAVs for the Navy's Surface Search and Control, emphasizing the use of Broad Area Maritime Surveillance (BAMS) UAVs and Vertical Take-Off UAVs (VTUAVs). Frantz [5] described two methods of autonomous control of UAV swarms: the first method uses a non-linear approach to minimize the error between the location of each particle and the target by accelerating particles through the search space until the target is found; the second method, uses a linear algorithm to determine the correct heading and maneuver the swarm toward the target at a constant velocity. Another use of the UAVs is explained by Schaeffer [6] describing how the integration of UAVs and Shotspotter, which is an acoustic gunshot detection system, can improve the situational awareness of an Expeditionary Strike Group. Teng [7] described the use of two types of UAVs, a tactical unmanned combat aerial vehicle (UCAV) equipped with high resolution sensors and electronics to support ground units, which acts as a mother-ship for smaller Killer UAVs equipped with a specific mission-dependent sensor or warhead.

While the above research focuses on general UAV employment, the open literature is largely lacking in unclassified sources to address unmanned systems as a counter UAV system. However, there are counter UAV systems, such as the ones developed by Alliant Techsystems (ATK) [8]. The first system consists of a projectile guided by an infrared proximity sensor that bursts in the vicinity of an enemy UAV and encapsulates the target with a rapidly expanding foam in order to disable the UAV without destroying it. This allows the possibility of recovering the UAV and its payload intact for intelligence purposes. The second proposed method is to use a directed energy weapon to shoot down the UAV.

Another company, QinetiQ and Sula Systems [9], has designed a low-cost countermeasure to tactical UAVs called the Cougar Interceptor. This system is an attack UAV that targets an adversary UAV to cause catastrophic structural damage to it.

The *Black Dart* field experimentation program [10], recognized as a venue for counter unmanned aerial vehicle experiments, is a joint agency demonstration which focuses on rapid development and implementation of counter UAV technology from readily-available commercial products. The organization leverages the "largest known database for sensor-weapons detection and response capabilities," including "various UAV signatures, sensors and defeat mechanisms to assess the current range at which such UAVs can be reasonably detected and determined state of the art, unconventional, near-term defeat options, including directed energy options" [10].

The research of swarm robotics derives much of its inspiration from natural systems. Social insects are known to coordinate their actions to accomplish tasks that are far beyond the capabilities of a single individual; termites build large and complex mounds, army ants organize impressive foraging raids, and teams of ants can collectively carry large prey [11]. Such coordination capabilities are known as swarm behavior, and there is a large body of literature describing different aspects or uses of this concept. Blum [12] describes swarm intelligence as a modern artificial intelligence discipline that is concerned with the design of multi-agent systems with applications, instead of a single sophisticated controller that governs the global behavior of the system. The swarm intelligence principle is based on many unsophisticated entities that cooperate in order to exhibit a desired behavior or task, for example, to saturate the air defense system of a ship. Other UAV swarm studies have been conducted by the Applied Physics Laboratory (APL) at the Johns Hopkins University [13], such as the communication necessary between UAVs to take the human out of the loop in terms of individual vehicle control. Another example of swarm in robotics, but in a small scale, is the *RoboCup* competition [14][15][16],

which is a worldwide tournament of robots playing soccer. The objective of *RoboCup* is "to promote robotics and AI research," where the artificial intelligence used here could be interpreted as a swarm behavior to accomplish one mission, that is, to score a goal. There is also a report written by Sahin [11] which aims to define what swarm robotics is, and proposes different uses of swarm robotics in different fields.

Swarming has been used to describe new tactics in military operations. Shannon [17] describe the use of swarm tactics in military operations such as the ones used on the Chechen Wars [2] and suggest the inclusion of swarming tactics in US Marine Corps doctrine. Arquilla and Ronfeldt [18] describe swarming as a "deliberately structured, coordinated, strategic way to strike from all directions, by means of a sustainable pulsing of force and/or fire, close-in as well as from stand-off positions." They also state that "developing a swarming force implies a radical changes in current military organizational structures, from command and control of line units to logistics." This is an important point because in addition to these new tactics, it is necessary to develop new systems like the ones described in this thesis.

Other studies of UAVs using agent based simulation has been performed over the years. For example, Ho [19] uses MANA (Map Aware Non-uniform Automata) to model ground robotics swarm on a search and detection mission in a semi-urban environment rigged with stationary improvised explosive devices. Lalis [20] use the MANA software to analyze the use of UAVs near Greek islands to detects illegal activities. Gill [21], uses MANA to evaluate a strike scenario that focuses on the coordination of the Navy Unmanned Combat Air System, the F/A-18 Super Hornet and the F-35C Lightning II. To the best of the author's knowledge, this thesis is the first work to use the Repast Simphony [3] simulation environment to simulate UAVs in an agent based environment. Given the open source nature of Repast, the open community provides ample documentation for how to implement multiple agents in three dimensional environments and how to create their respective behaviors. Further, the software is freely available and distributable, and the developed simulation model, built in Java, can be easily and rapidly deployed across operating systems for broader usage. These advantages over existing proprietary software suites lead to its use in this thesis.

[2]The First Chechen War, also known as the War in Chechnya, was a conflict between the Russian Federation and the Chechen Republic of Ichkeria, fought from December 1994 to August 1996. The Second Chechen War, a later phase better known as the War in the North Caucasus, was launched by the Russian Federation starting on August 26, 1999, in response to the Invasion of Dagestan by the Islamic International Peacekeeping Brigade (IIPB).

In terms of low cost UAVs, there are many open sources related to their design and control (see [22], [23]), where most are related to hobby radio controlled planes with some modifications for unmanned flight [24]. A report from Virginia Tech in 2003 investigates the feasibility of a low cost expendable UAV to used for reconnaissance missions where there is chance to lose the platform [25]. This report concludes with an estimated cost of production for a low-cost UAV and explains the assumptions considered in the study.

1.4 Scope, Limitations and Assumptions

1.4.1 Scope

This thesis focuses on examining the characteristics of the Blue UCAV swarm system by identifying important design factors that should be taken into account in the development of such a future system. Based on available and unclassified information, behaviors for each Red agent for use in the agent-based simulation software are developed. In addition, the proposed work presents and implements canonical Blue agent behaviors based on notional requirements to best address the Red threat. Rather than defining exact engineering specifications for a defensive swarm system, this thesis presents how the different factors of the design space relate to the proposed MOE as shown by experimental design and statistical analysis. The MOEs are quantitative measures that give some insight into how effectively a system is performing. For this thesis, the MOE is how many Red UCAVs are killed by the Blue UCAV system.

1.4.2 Limitations

Despite the operational relevance and importance of this topic, the lack of information in the open domain and at the unclassified level is formidable. Therefore, in some cases the author uses information pertaining to analogous systems for which data are openly available. However, the model and methodology presented in this thesis are designed to be flexible enough to accommodate more accurate and/or classified information as it becomes available.

1.4.3 Assumptions

For the simulation model, three types of agents are considered:

- Blue High Value Unit (HVU)
- Blue Unmanned Combat Aerial Vehicle (UCAV)
- Red Unmanned Combat Aerial Vehicle (UCAV)

Each of the three types of agents has its own customizable behavior. Each Red agent is assumed to be initially airborne, i.e., it is assumed that it has already been deployed from its base, and that sufficient time has already passed such that the Red agent is near the operating area of the Blue HVU. Since the goal of this thesis research is to study the design factors relevant to the Blue UCAV, the Red agent behaviors are relatively constrained. Blue UCAV agents are also subject to limiting assumptions, such as limited endurance, i.e., a time constraint on reaching an incoming Red UCAV. Also, the Blue UCAV is considered to operate as a small missile (i.e., no sophisticated flight dynamics), and targeting of a Red UCAV is assumed perfect and self-destructive, that is, a successful kill of a Red UCAV also requires the loss of the Blue UCAV. The Blue HVU is operating in a given patrol area and its objective is to remain in that area. The Blue HVU is assumed to accurately detect and immediately assign every incoming Red UCAV to a Blue UCAV. In addition, agent failures (e.g., mechanical or electrical) during the mission are not considered. However, stochastic behavior in each simulation run is generated by using probabilistic parameters associated with some variables as discussed in the next chapter.

1.4.4 Course of Study

Figure 1.1 graphically depicts the general flow of the research performed for this thesis. First, a literature review is executed to gather the necessary information to build each agent behavior. Then, the behaviors are implemented in the agent-based simulation environment. Prior to executing a design of experiment, the controllable and uncontrollable factors are identified. The design points obtained from the design of experiments are then simulated in the simulation model to produce data. Finally, the data is analyzed to obtain insights of the study.

1.5 Contribution of this thesis

The results from this thesis can be used as a starting point in a future design of a counter UCAV swarm system, because this thesis focuses on some of the design factors of such a system.

For future studies of UAV swarms, the presented model (see a screenshot from the simulation in Figure 1.2) can be used as a starting point for new scenarios or analyses of new factors that are not considered in the present study.

This thesis also serves as a guide for the simulation of UAVs or other systems in an agent-based simulation [26] [27] environment besides MANA, as other tools, such as Repast Simphony [3], may offer additional capabilities and/or more flexibility for future studies.

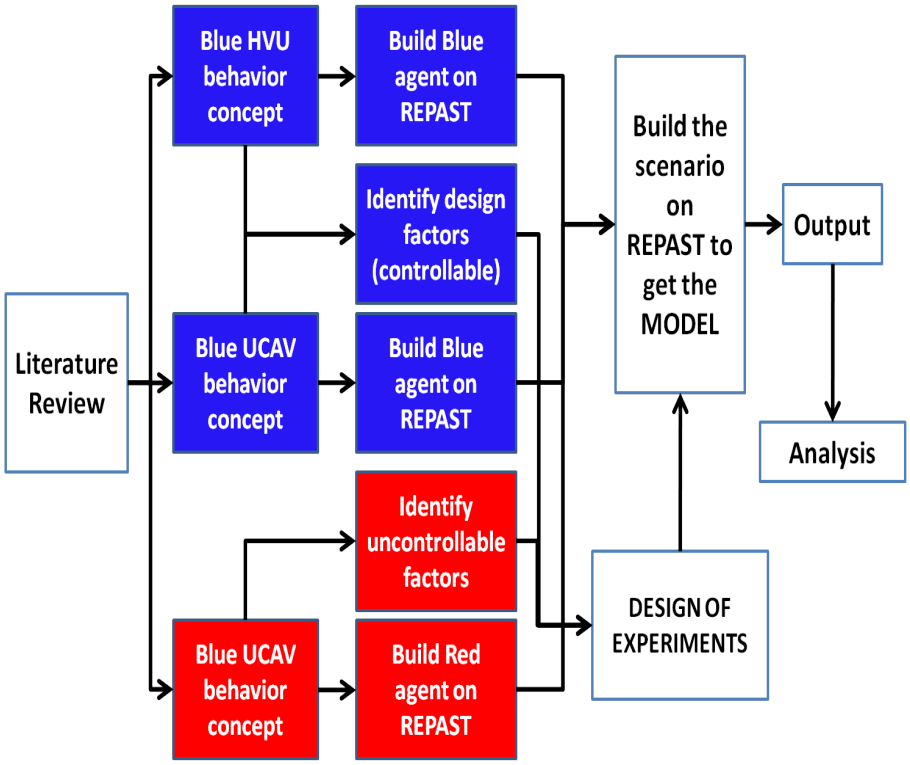

Figure 1.1: A graphical representation of the course of study for this thesis

1.6 Organization of the thesis

Having presented a review of the literature above, Chapter 2 describes the general scenario as well as the behaviors of each agent type (i.e., Red UCAV, Blue UCAV, and Blue HVU agents). Then in Chapter 3, the relevant factors to be analyzed are described and a design of experiment proposed. Analysis addressing significant factors is performed and presented in Chapter 4, including a new design of experiment to refine a prediction model that optimizes the performance of the Blue UCAV system. The thesis concludes with Chapter 5, which summarizes results and presents recommendations and also provides avenues for future studies.

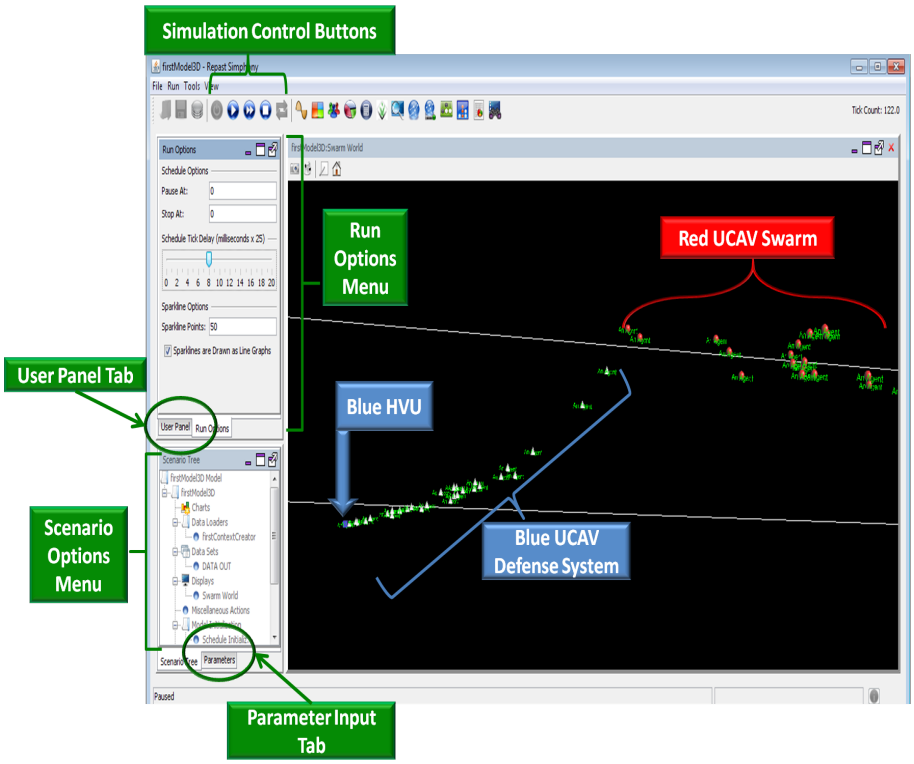

Figure 1.2: Screen shot of the implemented agent-based simulation model

CHAPTER 2:
Model Formulation

In this chapter, the operationally relevant context is provided as motivation for this research, and each agent type and behavior implemented in the developed simulation model are described. Further, a brief discussion on two alternative approaches initially considered is presented and contrasted with the agent-based simulation approach ultimately proposed by this thesis.

2.1 Real World Scenario

To be able to create a realistic scenario model, the author's simulation is based on the following situation:

> *Consider a US or allied force deployed in international open waters to patrol a certain region. This force is equipped with a new defense system against a UCAV swarm threat. The defense system consists of a large number of small UCAVs each equipped with an explosive warhead capable of destroying an enemy UCAV in flight. On the other hand, the enemy force is known to have a fleet of UCAVs that operates as a swarm to saturate the air defense system of an adversary's force and then to engage with ballistic missiles to destroy it. The UCAV swarm can be deployed either from a ground based platform or sea based platform.*

Consideration must be given in the above situation to the fact that the number and characteristics of enemy UCAVs that make up the swarm is unknown. For the purpose of the proposed agent-based simulation model, the US or allied force is represented as a single high value unit (HVU), assumed capable of deploying the future defense system against the UCAV swarm. The scenario is described in Figure 2.1, where it is possible to observe that the Red UCAV agent has three different flight modes: Loiter, when the Red UCAV is flying at its minimum speed searching for a target; Approach, when the Red UCAV has detected the target and flies towards it increasing the speed but keeping a fixed altitude; and Attack, when the Red UCAV dives towards the target increasing its speed to hinder countermeasures and produce the most damage. To separate the three modes is important because there are different factors associated with each mode. It is also possible to observe that the Blue HVU is the platform where the Blue UCAVs are launched against the incoming Red UCAVs.

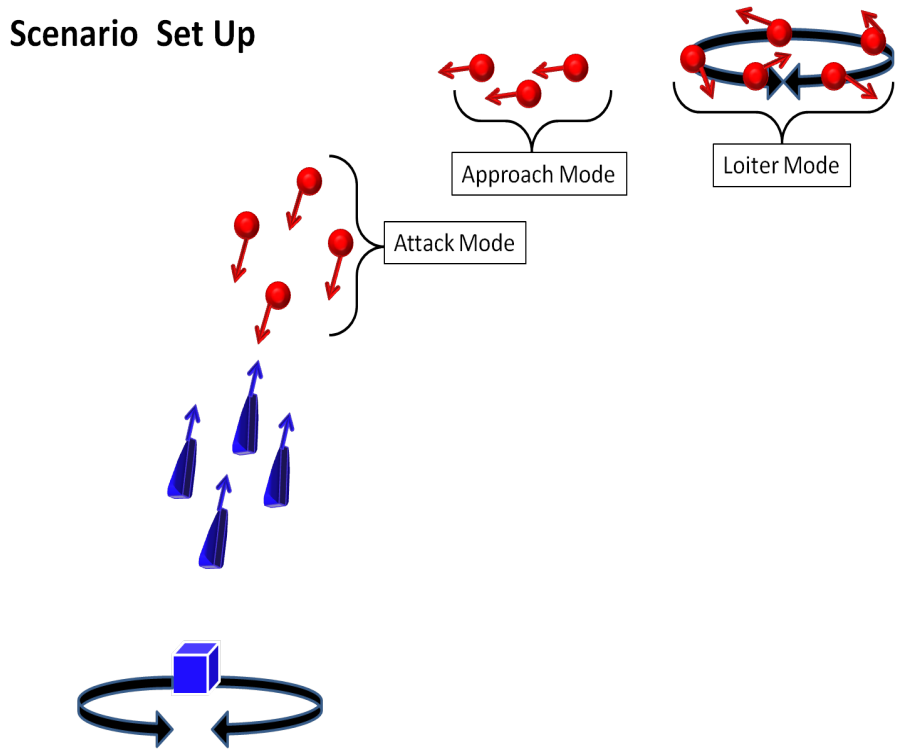

Figure 2.1: The Blue cube represent the Blue HVU patrolling in a certain area. The Blue triangles represent the Blue UCAV defense system. The Red spheres represent the Red UCAV swarm.

This thesis incorporates publicly available information in an iterative fashion, such that once the first simulation model is constructed, a new literature review is performed to make sure that the simulation model reflects the real world situation. This process is repeated several times in order to get the final model with reasonably realistic characteristics. Figure 2.2 provides a diagram of the general flow of the development of this work.

2.2 Agent Descriptions

Based on the information gathered from the literature review and the real world scenario description, each agent is described with its own behavior. To distinguish the US or allied force from the enemy force a color is assign to each force, i.e., Blue for the US or allied force and Red for the enemy.

2.2.1 Blue High Value Unit (Blue HVU)

The HVU is able to detect the incoming Red UCAVs and assign and launch the Blue UCAVs. The behavior of the HVU is simple. It needs to patrol in a certain area. However, it needs to detect, assign, launch, and keep track of all airborne UCAVs (Red or Blue). In Figure 2.3, it

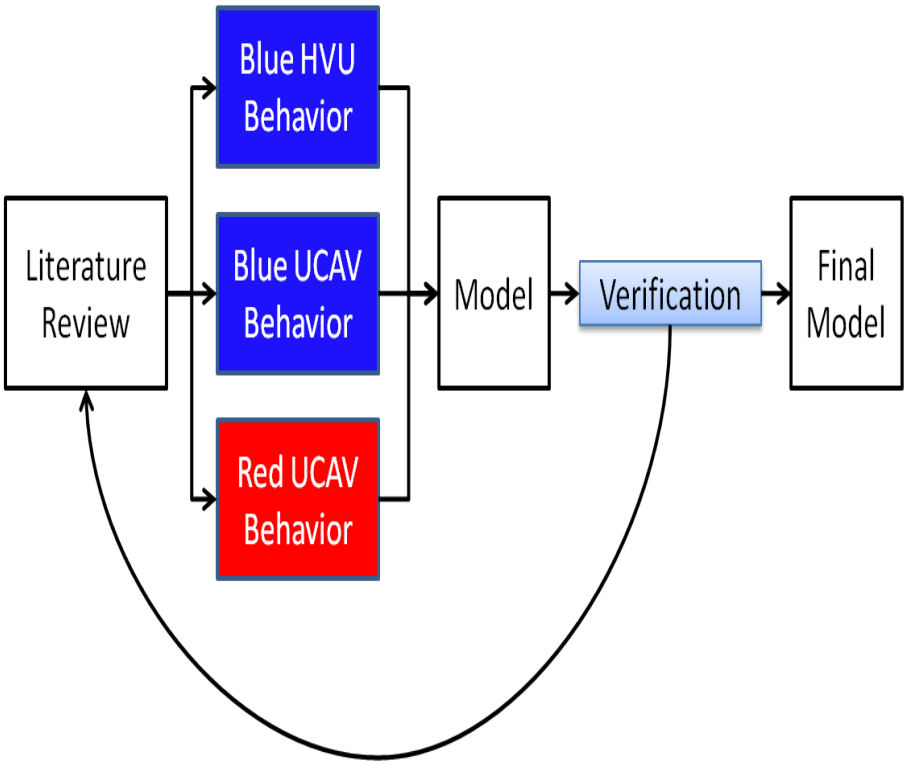

Figure 2.2: This figure shows the process to obtain the final model. First, a literature review was performed in order to understand and define the behavior for each agent. Then, a model was implemented with the desire behavior for each agent. In order to verify if the model if the model does what it was design for, a second literature review was performed. In this case, the second literature review was focus on details such as altitude at which real UAV flies, check the endurance values and actual Blast ranges. When the model fits with all the requirements from the literature review and the desire behavior for the Blue UCAV defense system, the final model was obtained.

is possible to see the behavior in a flow diagram. Notice that all the elements associated with keeping track of the information from the Red and Blue UCAVs are not explicitly identified but rather assumed available in real-time. At the beginning of the simulation the HVU is operating in a certain patrol area. Then at a certain distance it will start detecting the incoming Red UCAV. When the distance to the Red UCAV is less than the *Launching Range*, the HVU will launch and assign a Blue UCAV to the incoming Red UCAV. The HVU will keep track if the assigned Red UCAV has been destroyed or not. If one of the Red UCAV reaches the HVU, it will damage the HVU, and if the damage is sufficient enough to disable the HVU the simulation will end, and it will be assumed that the Red UCAV swarm has accomplished its mission.

2.2.2 Blue Unmanned Combat Aerial Vehicle (Blue UCAV)

The Blue UCAV is able to move in a three-dimensional space. Each Blue UCAV agent is launched upon request from the HVU. It possesses a means onboard to estimate and predict the position of the assigned Red UCAV, and also to measure the range to target so that the

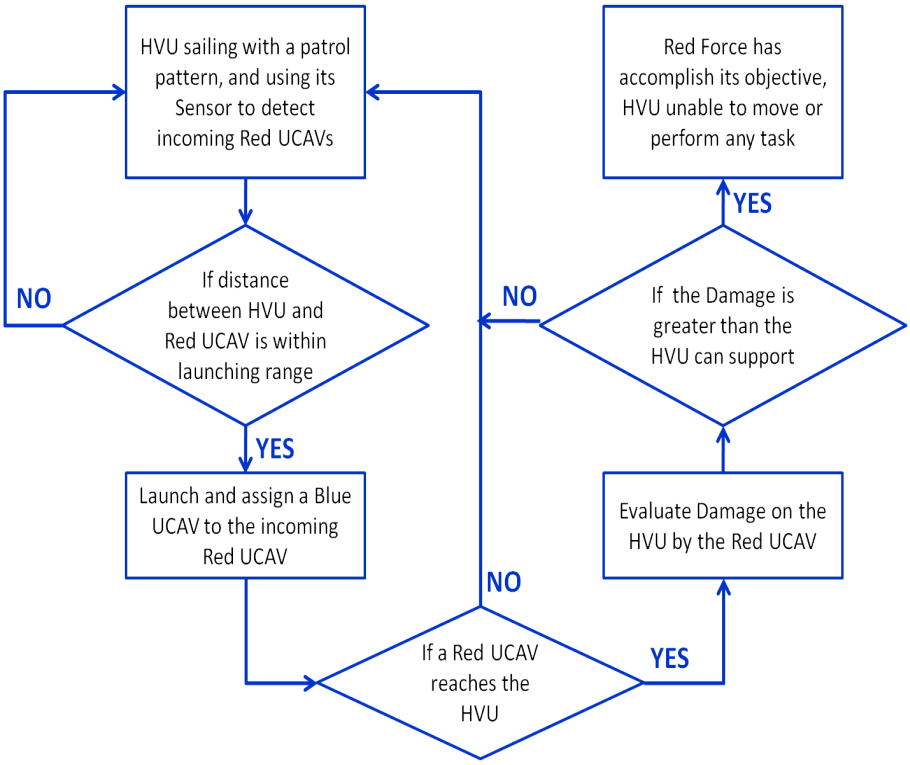

Figure 2.3: This diagram have two types of blocks, the rectangles that represent actions of the HVU agent; and decision blocks, represented by the parallelogram

Blue UCAV can adjust its own speed to close on the assigned Red agent and then activate the warhead to explode to destroy it. Each Blue UCAV launched have only one Red UCAV assigned to it to engage. In Figure 2.4 the behavior diagram of the Blue UCAV can be seen. After it is launched, the Blue UCAV needs to predict the future position of the assigned Red UCAV. This process is performed every time step during the simulation. Then it needs to check the distance to the assigned Red UCAV. If the distance to the assigned Red UCAV is within the *Blast Range*, the Blue UCAV detonates, and it have a probability of kill associated with that explosion to determine if the assigned Red UCAV is killed or not.

2.2.3 Red Unmanned Combat Aerial Vehicle (Red UCAV)

The Red UCAV is also able to move in a three-dimensional space. According to the literature review, an analogous operational system to the Red UCAV studied in this thesis is the IAI *Harop*, i.e., *Harpy*, platform, and its publicly available characteristics serve as a basis of the proposed behavior model. The Red UCAV will have three different modes of operation. The Loiter Mode, the first mode of operation, is when the Red UCAV is searching for a target, i.e., the HVU. During this mode, the Red agent will have a fixed altitude and will move in a certain

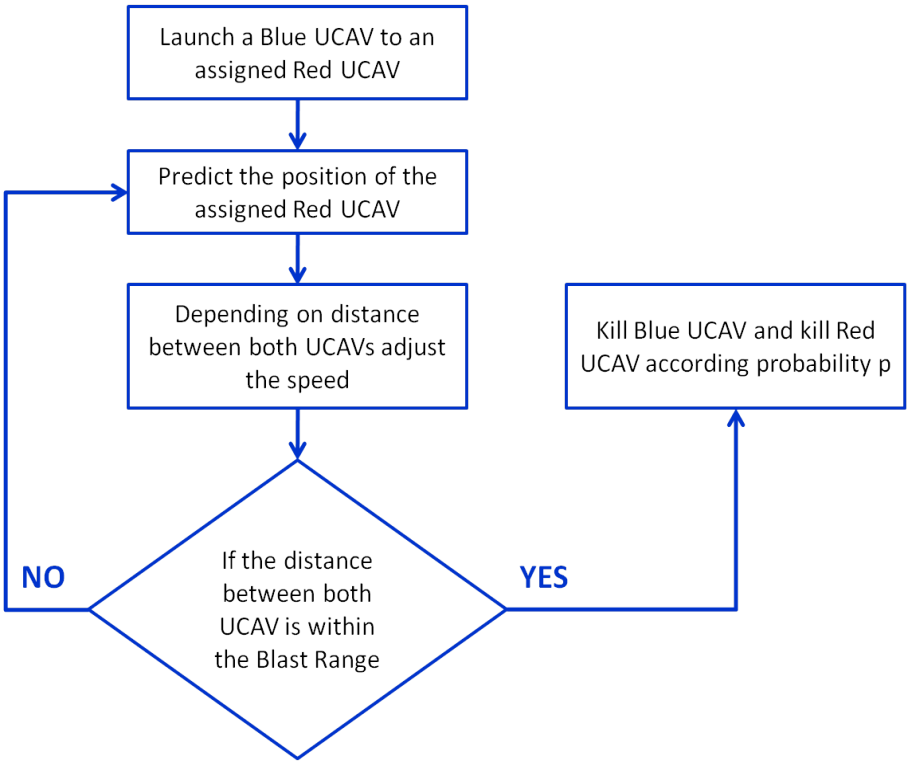

Figure 2.4: In this figure, the rectangles represent actions performed by the Blue UCAV in every time step, and the parallelograms represent a decision, which is like the distance sensor to activate the warhead of the Blue UCAV to explodes in the proximity of the assigned Red UCAV.

loiter area defined in the scenario. The second mode is the Approach Mode, and is when the Red UCAV detects the HVU and approaches it, increasing its speed but keeping the same altitude. The last mode is the Attack Mode, when the Red UCAV reaches a certain angle between the vertical of the HVU and the HVU (Figure 2.5) and dives toward the HVU increasing its speed to perform a kamikaze type of attack. In Figure 2.6 the behavior diagram for the Red UCAV illustrates the transitions from one mode to another.

2.3 Simulation Model

Now that the behaviors of each agent have been described, it is necessary to describe the simulation environment in which the agent are going to interact. As was described in Chapter 1, Repast Simphony is used to construct and perform the simulation. This software is a JAVA-based program; therefore, the behaviors described in the above sections are written in the JAVA language.[3]

[3]The source code of the behaviors can be found in the Appendix B.

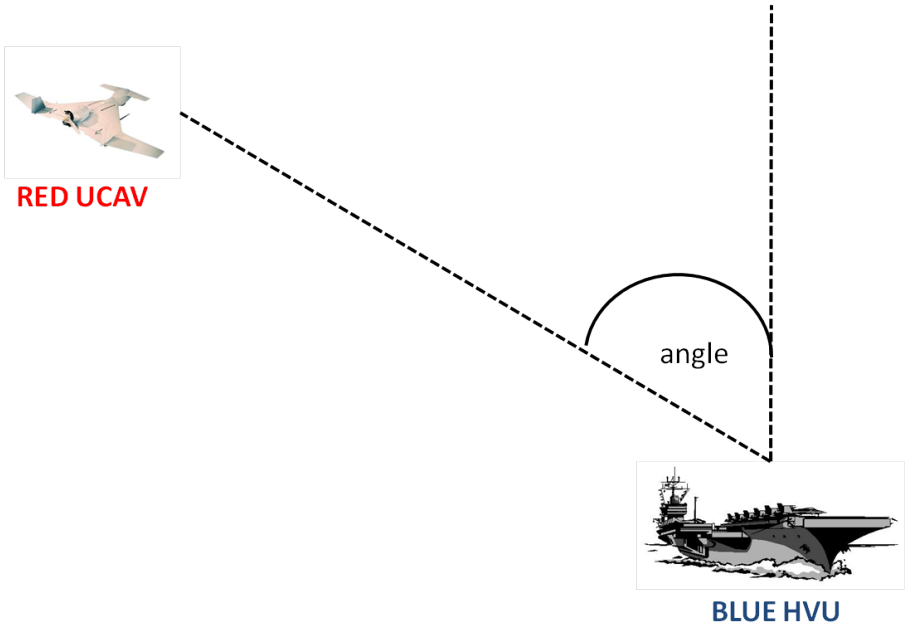

Figure 2.5: Angle between Red UCAV and HVU

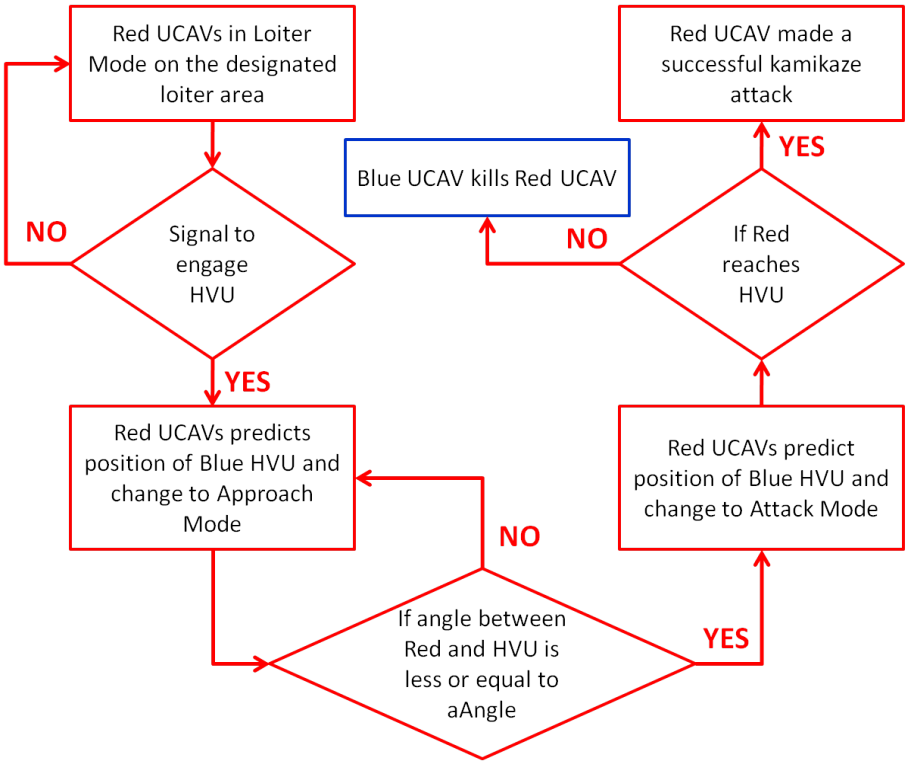

Figure 2.6: Red UCAV behavior diagram

One of the features of Repast is the possibility to create a three-dimensional space. The orientation of the axis coordinates as defined in the simulation can be visualized in Figure 2.7.

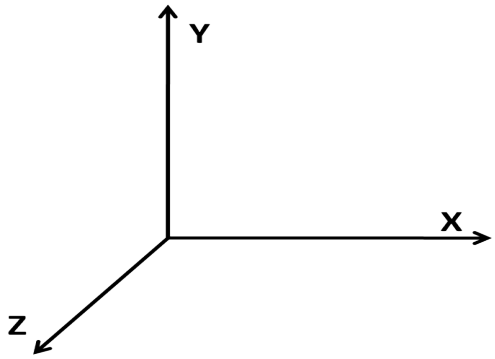

Figure 2.7: Axis coordinate orientation

The scenario built in Repast is a box of 3000 units on the X axis, 90 units on the Y axis and 3000 units on the Z axis. For purposes of the simulation, one unit in one axis represents 100 meters in the real world. This means that a surface of 300 km^2 and an air space of 9,000 meters (approximately 30,000 feet), which is the altitude of a MALE (medium altitude up to 30,000 feet, long endurance more than 120 nm) UAV [28]. This box represents the area of patrol of the HVU and the airspace above it where the aerial agents (i.e., Blue and Red UCAVs) can operate. A special space of this box is reserved for the loiter area of the Red UCAVs. This loiter area is defined in the source code of the simulation, and is located as far away as possible from the position of the HVU within the defined box. In Repast, in order to run the simulation an internal clock to schedule the movement of each agent is used. This clock increases one unit or one step at a time. For the purpose of simulation, every time step is considered as one minute in the real world.

2.3.1 Agents in the Repast Simulation Environment

Repast uses a different Java class for defining each agent. Every agent needs at least four **methods** in the Java language to run the simulation:

- The constructor method, where the factors are defined for each agent.
- The `step()` method, where the different tasks are executed. Every task, depending on the agent, can have a separate method.

- The `move()` method, where the agent is told how and where to move.
- The `die()` method, where the agent is removed from the simulation.

When an agent is created in the simulated environment, the factors from each agent are initialized in the constructor method. A brief definition of each factor is given below, though a more detailed description of these factors is given in Chapter 3. Also, any additional methods in support of the final agent-based simulation model are also described below.

Blue HVU Factors and extra methods
When the Blue HVU is created the following factors are initialized:

px, py, pz are the position on each coordinate. py in this case will always be equal to zero.

Speed is the speed at which the HVU moves in the patrol area. It is measured in knots.

Energy is a variable to identify the level of damage of the HVU.

Detection range is the distance at which the HVU detects the incoming Red UCAVs. It is measured in nautical miles.

Blue per Red is the number of Blue UCAVs that are going to be launched per each incoming Red UCAV. It is measured in Blue UCAVs.

Critical Red is the number of Red UCAV attacks that the HVU can receive before unable to perform any task, i.e., considered destroyed. It is measured in Red UCAVs.

Red Array is an array to store the information of all the Red UCAVs.

Distance Array is an array to store the distance from the HVU to each Red UCAV.

Engage Array is an array to keep track of which of the Red UCAVs have been engaged.

Kill Array is an array to keep track of which of the Red UCAVs have been killed.

For the purposes of the simulation the detection range is also the launching range which is the distance at which the Blue UCAVs are launched. Note that the launching range, however, depends on the endurance and speed of the Blue UCAV and also the distance of the assigned Red UCAV. Besides the four main methods, the HVU class has methods to return the results of

the simulation, such as the number of Blue UCAVs that were launched, the number of Red hits to the HVU, the number of Red UCAVs that are in the area and a method to reduce energy until the HVU is killed by the specified number of hits by the Red UCAVs. When the HVU is killed or dies, the simulation will end, because this will mean that the Blue UCAV defense system has failed. However, the number of Red UCAVs killed before the HVU dies will be recorded and saved as an output of the simulation.

Blue UCAV Factors

These are the factors that are initialized when the HVU launches each Blue UCAV:

px, py, pz are the positions in each coordinate.

vx, vy, vz are the velocities in the three different axes.

Current Red is a variable of the assigned Red UCAV to the new Blue UCAV.

Red Point is a variable to store the position of the assigned Red UCAV.

Blue Flag is a boolean variable to indicate if the assigned Red UCAV was killed or not.

Blast Range is the lethal range when the Blue UCAV explodes near the Red UCAV. It is measured in (fractions of) nautical miles.

Blue Speed is the maximum speed at which the Blue UCAV flies. It is measured in knots.

Blue Endurance is the lifetime of the Blue UCAV from being launched. It is measured in hours.

When the Red UCAV is within the blast range of the Blue UCAV, there is a probability of kill associated with the explosion. This probability of kill is fixed in the code but can be changed at any time, if necessary. The value chosen for the probability of kill is 0.85 and it was selected based on information garnered from the literature review. Another consideration for the Blue UCAV is that the launching range, described in the Blue HVU, depends on the endurance and Blue UCAV speed and distance of the assigned Red UCAV, because if the Blue UCAV is launched when the Red UCAV is too far away, it may run out of fuel or battery before reaching its target. That is why, before the Blue UCAV is launched, the maximum range (i.e., endurance times Blue UCAV speed) is compared with the launching range and the distance to the assigned Red UCAV.

The method created to adjust the behavior of the Blue UCAV to reality is a noise method, that induces error to the true position of the Red UCAV. However, the first idea was to create a predictor method that would use the current position of the Red UCAV and its heading and speed to move the Blue UCAV to the future position of the Red. The problem with this is that Repast, in every time step, performs all the calculations, tasks and moves for each agent; therefore, the Blue UCAVs is always ahead of the Red UCAVs. For this reason, the noise method is used instead of the predictor method.

Red UCAV Factors

When the swarm of Red UCAVs is created in the simulation environment, the following factors are initialize:

px, py, pz are the positions in each coordinate.

vx, vy, vz are the velocities in the three different axes.

Loiter Speed is the speed of the Red UCAV in Loiter Mode. It is measured in knots.

Approach Speed is the speed of the Red UCAV in Approach Mode. It is measured in knots.

Attack Speed is the speed of the Red UCAV in Attack Mode. It is measured in knots.

Current Speed is a variable to keep track of the current speed during the simulation. It is measured in knots.

Time is a variable to keep track of the time of the simulation.

aAngle is the attack angle that the Red UCAV has to reach to change from Approach to Attack Mode. It is measured in degrees.

Engage is a variable to set the (random) time when the Red UCAV detects the Blue HVU. It is measured in hours.

From the above, it can be seen that Loiter, Approach and Attack Speed and aAngle are the same value for all the Red UCAVs. All the other factors are independent for every Red UCAV in the swarm.

The extra methods in the Red UCAV class are basically setters and getters, to either set a value or get a specific value. Transitions by the Red UCAV from Loiter to Approach modes is determined by a uniform distribution, though in future work, alternative distributions may be studied. This distribution dictates when each Red UCAV switches modes, which is assumed identically and independently distributed for each agent within the simulation time window of time steps 15 to 115.

2.4 Development of the Model

Several trials and runs were performed to obtain the final model. The initial model was to create a Red agent that flew towards the HVU agent. The second model included the Blue UCAV agent to finally achieve the definitive model for this thesis, which may be improved upon in a future study. The model development plan and work flow used over the course of this thesis is shown in Figure 2.8.[4]

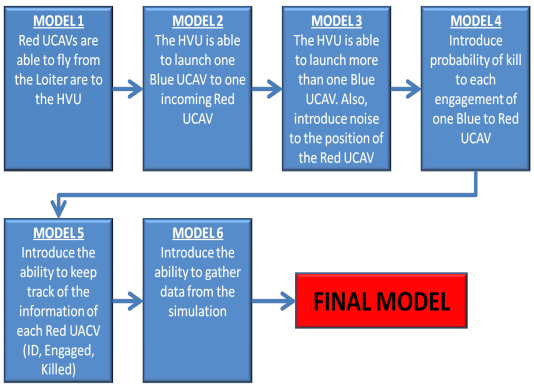

Figure 2.8: Model Development

2.5 Different Modeling Approaches

In addition to the presented agent-based simulation methods, two different approaches were explored in order to understand the problem using mathematical tools. However, these two approaches, at least at the preliminary levels at which they were studied, did not provide the insights that the author was seeking to determine the significant factors.

The first approach was to analyze the scenario using combat modeling [29], specifically Lanchester Models [30]. Lanchester Models are systems of ordinary differential equations where the state variables represent numbers of surviving combat units. During the height of World War

[4]The final model source code can be found in the Appendix B.

I (1916), Frederick Lanchester applied these equations to aerial combat. Two special cases of Lanchesters equations exist: the "square law" and the "linear law."

The "linear law," also called Unaimed Fire, can be applied with the assumption that one man can fight exactly one other man at a time, and if identical weapons are being used, the end of the battle will be the difference between the larger and smaller force. The equations that model the "linear law" uses are the following:

$$\frac{dB}{dt} = \dot{B} = -\beta BR, \qquad B > 0$$
$$\frac{dR}{dt} = \dot{R} = -\alpha RB, \qquad R > 0$$

where

$\dot{B}$ = ratio at which blue force decreases

$\dot{R}$ = ratio at which red force decreases

α = attrition coefficient at which blue kills red force

β = attrition coefficient at which red kills blue force

B = initial number of blue forces

R = initial number of red forces

The square law (a.k.a. Aimed Fire) represents a situation where attrition to each side is proportional to the number of units remaining on the other, and there are no reinforcements. The ordinary differential equations that model the "square law" are the following:

$$\frac{dB}{dt} = \dot{B} = -\beta R \qquad B > 0$$
$$\frac{dR}{dt} = \dot{R} = -\alpha B \qquad R > 0$$

These equations can be solved explicitly for B and R as a function of time. However, it is simple to eliminate time by dividing the second equation by the first where $\frac{dR}{dB} = \frac{\alpha}{\beta}\frac{B}{R}$ can be solved as: $\beta(R^2 - R_0^2) = \alpha(B^2 - B_0^2)$. To run an analytical model, it is possible to use the

following formulas and obtain initial solutions for the UCAV problem,

$$B(t) = B_0 \cosh(\sqrt{\alpha\beta}t) - R_0\sqrt{\frac{\beta}{\alpha}} \sinh(\sqrt{\alpha\beta}t)$$

$$R(t) = R_0 \cosh(\sqrt{\alpha\beta}t) - B_0\sqrt{\frac{\alpha}{\beta}} \sinh(\sqrt{\alpha\beta}t)$$

where

$$\begin{aligned} B(t) &= \text{is the number of blue UCAVs at time t} \\ R(t) &= \text{is the number of red UCAVs at time t} \\ B_0 &= \text{initial number of blue UCAVs at time equal zero} \\ R_0 &= \text{initial number of red UCAVs at time equal zero} \\ \alpha &= \text{attrition coefficient at which blue kills red} \\ \beta &= \text{attrition coefficient at which red kills blue} \end{aligned}$$

Using this approach allows focus to be placed just when the Red UCAVs are in the attack mode and Blue UCAVs are launched against those particular Red UCAVs as described in Figure 2.9. Therefore, this approach only focuses on one part of the main problem and it does not deal with the Red UCAVs that are in the other two modes.

The possible results that can be obtained from this approach, though, are described in Figure 2.10.

The second approach to analyze the scenario is to use a binomial modeling approach [31]. It can be assumed that an engagement between a Red and Blue UCAV is a binomial experiment for the following reasons:

- This experiment consists of n repeated trials (each trial is an engagement).
- Each trial can result in just two possible outcomes (Kill the Red UCAV or have the Red UCAV survive the engagement).
- The trials or engagements are independent; that is, the outcome of one trial does not affect the outcome of other trials.

- The probability of success is the same in every trial or engagement. However, it will be assumed that the probability of success will be different for each engagement, because it will depend on different factors such as the mode in which the Red UCAV is flying, the distance at which the Blue UCAV explodes from the Red UCAV, and the accuracy of the Blue UCAV's estimate of the Red UCAV position.

Under the assumptions of a binomial experiment, an engagement between a Red and Blue UCAV can be defined mathematically. Better said, the probability that a Red UCAV survives a Blue engagement is:

$$P(\text{One Red survives}) = (1 - P_{\text{kill}})^B \tag{2.1}$$

where

$$P_{\text{kill}} = \text{the probability that a Blue UCAV kills a Red UCAV}$$
$$B = \text{the number of Blue UCAVs engaging the same Red UCAV}$$

The problem with this approach is a lack of the time factor, which is important for determining in which mode the Red UCAV is engaged. Also, it does not address other factors such as the launching distance and the speed of the Blue UCAV. It must be remembered that one of the objectives of this thesis is to identify the design factors, and Blue UCAV speed is one of them.

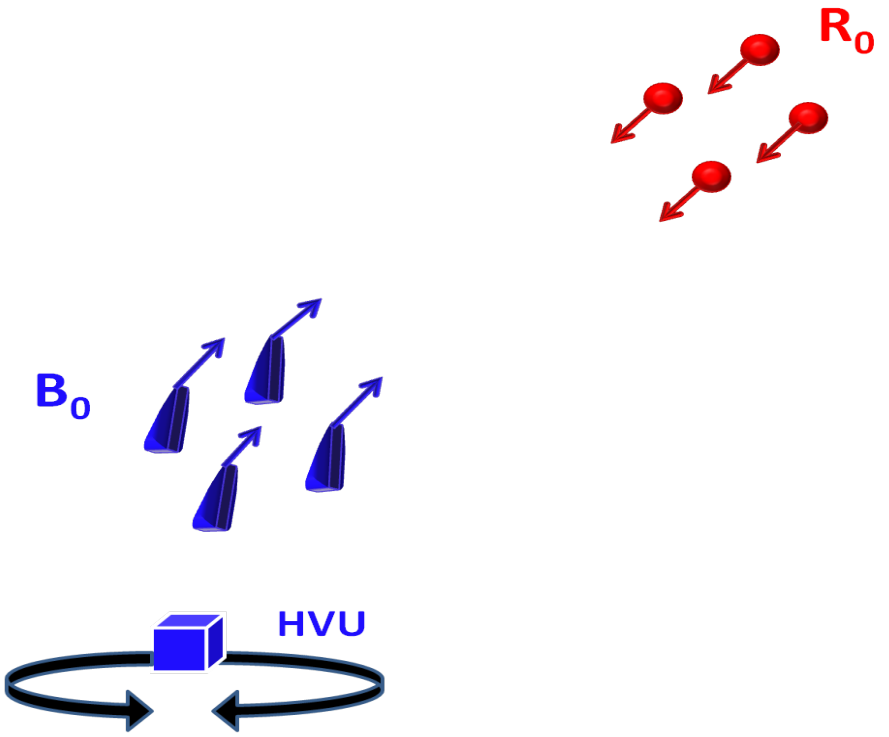

Figure 2.9: First Analysis Approach

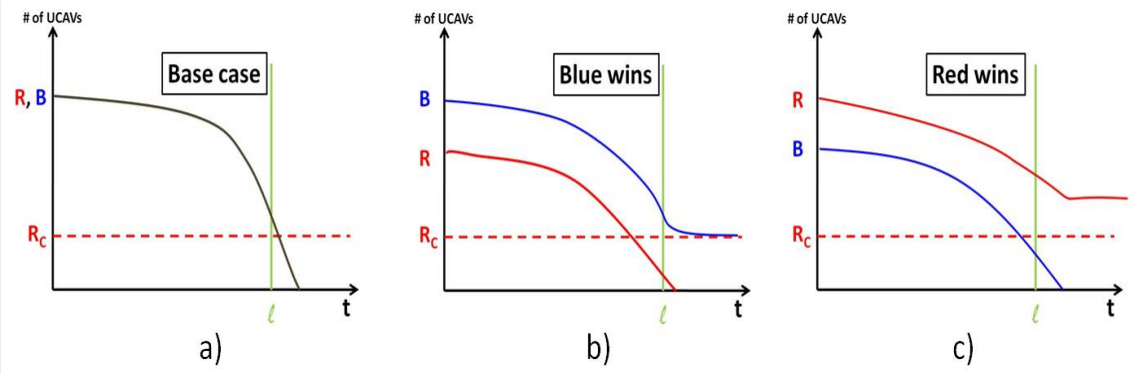

Figure 2.10: Results using first Approach, a) is the situation when the parameters are equal, $B_0 = R_0$ and $\alpha = \beta$. The point l at the time axis is the time it takes for one Red UCAV to reach the HVU. b) is the situation when the parameters are the following: $B_0 > R_0$ and the attrition coefficients are similar. At this point it is important to note this graph represent the "square law," where the rate of change of one force is equal to the square of the opposite force. c) is when the initial number of Reds are greater than the initial number of Blues and the attrition coefficient is similar.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Experimental Design

In this chapter, all the factors that are involved in the simulation, including both controllable and uncontrollable ones and the range of relevant values to be examined are identified. Description of the design of experiments (DOE) methodology and the reasons for the selected approaches are provided.

3.1 List of Factors

The factors that affect the behavior and therefore the outcomes or results of the simulation can be divided into two groups, namely controllable and uncontrollable. The controllable factors are the ones related to the design of the individual Blue UCAV such as its speed, blast range, and endurance, which can be defined as engineering specifications. On the other hand, the uncontrollable factors are those for which there is not enough information or are governed by external factors, such as the number of incoming Red UCAVs and their respective endurance.

3.1.1 Controllable Factors

All these factors are related to elements which pertain to the design of the Blue UCAV system and the Blue HVU. All of the information described below is obtained from open sources.

Blue UCAV speed

This factor, for the purpose of simulation, is measured in meters per second and is a continuous factor. It determines how far from the HVU the Blue and Red UCAV are going to engage. This is relevant because one might expect that it is more likely that the Red UCAV is interdicted successfully by the Blue UCAV during its Approach phase rather than in the Attack phase (refer to the scenario model description in the previous chapter) because the Red is slower and flies at a fixed altitude in the Approach mode. Open sources and other available literature provide a wide range of speeds for the different kinds of UAVs or UCAVs, starting at 18 m/s or 35 knots for the Wasp III [32] and reaching 0.8 Mach for the CORMORANT UAV [33]. Given the focus of this thesis on considering small UCAVs with limited engine size and power, the investigated speed range starts at 18 m/s or 35 knots and ends at 44.7 m/s or 87 knots. Figure 3.1 provides a summary from open sources of the speed of various UAVs and UCAVs. Those in blue correspond to small UAVs which are relevant and analogous to the proposed system.

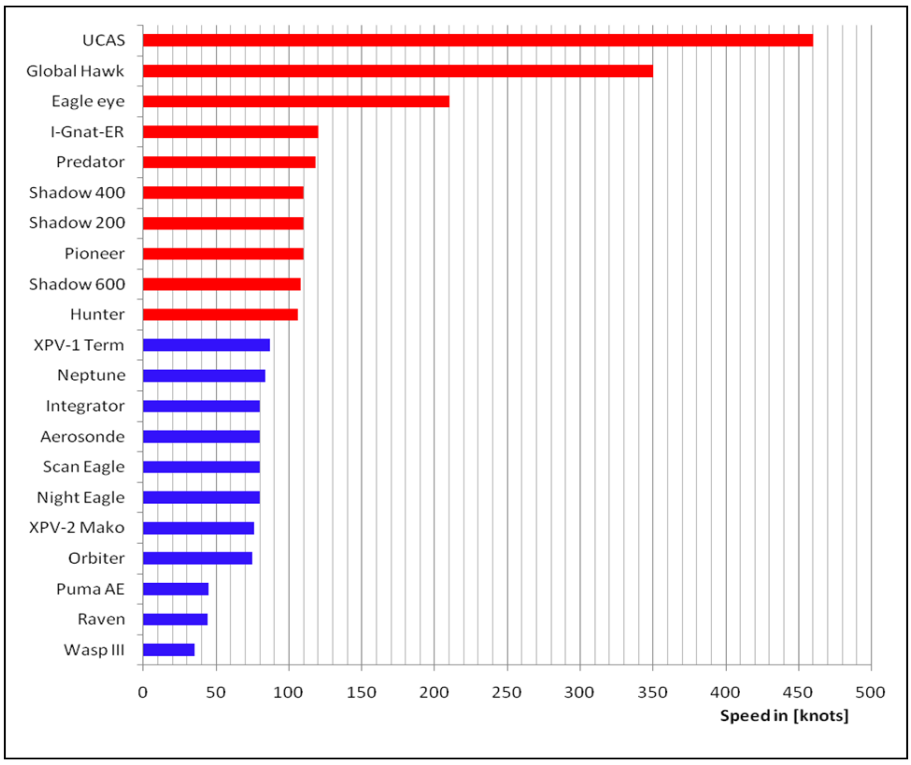

Figure 3.1: Speeds for existing UAV and UCAV systems, compiled from the open literature (see [33], [34], [35], [36], and [37]). The Blue blocks are the speed of the small size UAVs, and the Red blocks are the speed of the mid/large size UAVs.

Preliminary investigation via a few exploratory simulation runs using speeds in the slower end of the proposed range shows that the the slower Blue UCAV is unable to reach and successfully engage the faster incoming Red UCAV (described later in this section), which is logical because as the Blue approaches Red, the Red UCAV uses its faster speed to leave the Blue UCAV behind. This fact leads to the requirement that the Blue UCAV speed be comparable to the Red approach speed. Therefore, the interval of the factor levels is revised to range from 45 m/s (87 knots) to 54 m/s (105 knots).

Blast Range

Blast Range is the lethal range at which the Blue UCAV explodes in the proximity of the Red UCAV and destroys the Red UCAV. This factor depends on the amount of explosive that is loaded on the Blue UCAV. It is a continuous factor and is measured in meters. The purpose of this factor is to determine an effective Blast Range. Determining the best type of explosive material is beyond the scope of this research. A simple assumption is made that if the Blast Range increases, the payload in terms of weight of the Blue UCAV also increases. There is not much information available in open sources related to Blast Range pertaining to UCAV

applications. However, information about Anti Air (AA) missiles and their effective blast ranges with the respective amounts of explosive are available. In Figure 3.2 three different missiles with different amounts (in kilograms) of explosives and different blast ranges are shown (See [38], [39], and [40]).

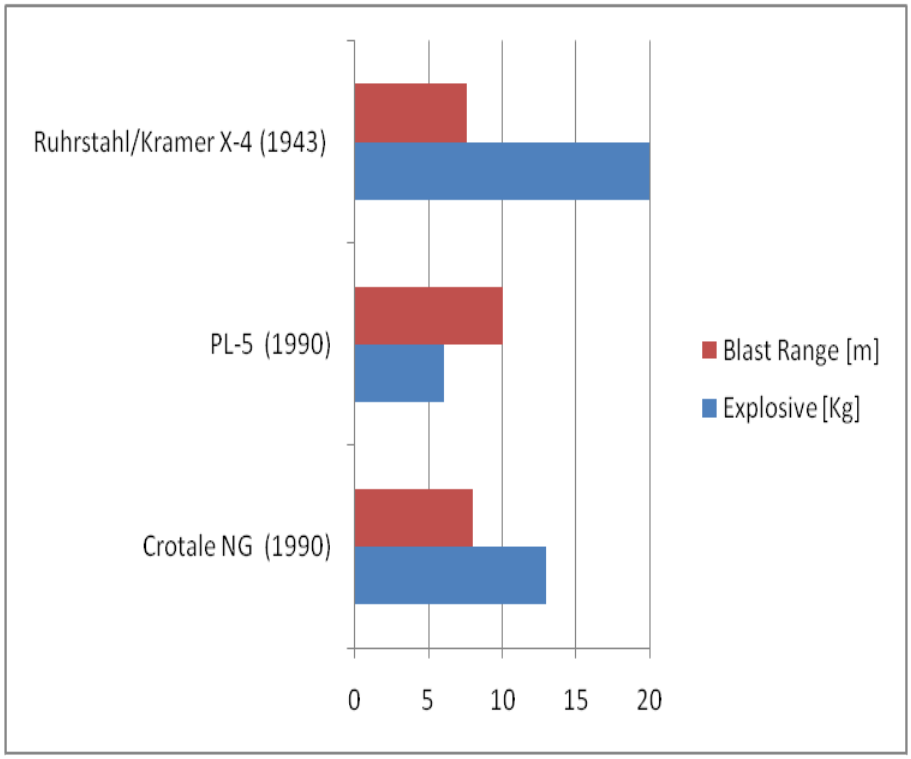

Figure 3.2: Representative anti-air missile blast range and explosive material quantity compiled from the open literature (See [38], [39], and [40])

The exemplar relationship between the amount of explosive and blast range is the one represented by the AA missile PL-5, since with less explosive a bigger blast range can be achieved. Given that the information available is about systems from the early 1990s, it is reasonable to think that a larger blast range with a smaller quantity of explosive can be achieved. Conservative approximation and the scope of this study lead to an assumed range of values for the Blast Range factor to be between 10 and 20 meters.

Blue UCAV Endurance

Endurance is a time measure and its units are seconds for the purpose of simulation[5]. Therefore, endurance can be considered as the lifetime of the Blue UCAV. It is a continuous factor and

[5]The need for resolutions on the order of seconds in simulation arises given the high speed nature (i.e., sub-minute) of the air-to-air interdiction.

represents the time that it takes for a Blue UCAV from its launch until it runs out of fuel or battery (in the case that the Blue UCAV uses an electric engine). As a reference to obtain a range of values, the information available from other UAV and UCAV systems is used. Similar to the Blast Range, an increase on Endurance is assumed to be an increase in payload weight of the Blue UCAV (i.e., more fuel or bigger battery). From Figure 3.3, the Endurance time from other UAV or UCAV systems can be seen. The endurance time of the Global Hawk is more than 30 hours, but this is a large UAV. This study will focus on the comparable small UAVs indicated by blue color in Figure 3.3.

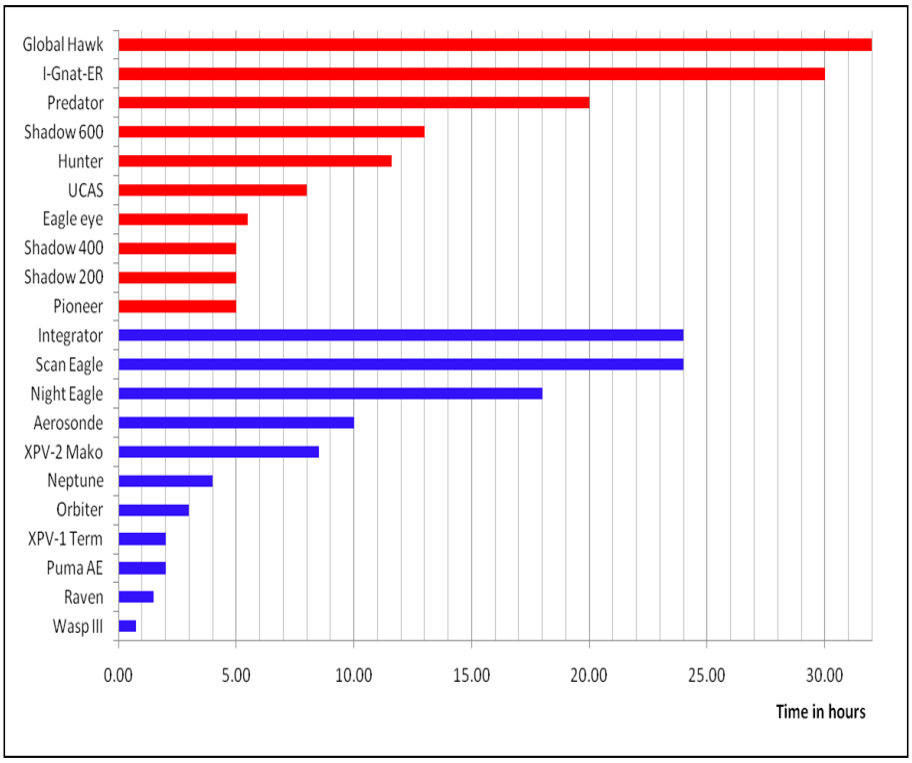

Figure 3.3: Endurance times for various UAVs or UCAVs systems. See [33], [34], [35], [36], [37], and [32]. The Blue blocks are the Endurance of the small size UVAs, and the Red blocks are the Endurance of the mid/big size UAVs.

From Figure 3.3 it can be determined that the values for Endurance in the proposed simulation experiments are between $2,000$ and $4,000$ seconds or equivalently 33 to 66 minutes.

Critical Number of Red UCAVs

This is a continuous integer factor and reflects the maximum number of Red UCAV strikes that the Blue HVU is able to manage and/or withstand without the employment of the Blue UCAV defense system. This baseline reflects the HVU's organic ability to destroy incoming Red UCAVs using Anti-Air (AA) weapons system such as the Close-In-Weapons-System

(CIWS) or AA missiles (or even future weapons such as directed energy), or the sum of the HVU speed and armor that allows the HVU to absorb a Red impact. It is assume that the lower limit for this factor is one if none of the HVU's AA defenses are used (i.e., a single hit by a Red UCAV destroys the Blue HVU) or passive defenses (e.g., evasion speed or armor) cannot be improve. For the upper limit of this factor, a value of 15 assumes that the effective combined defense by speed, armor, and AA systems are fully maximized and utilized.

Detection Range

Detection Range is the distance at which the Blue HVU sensors detect the incoming Red UCAVs. This, too, is a continuous factor and is measured in meters. The performance of these sensors, such as a phased array radar used by US Navy ships are dictated by numerous factors, including the size of the contact. Therefore, a small target such as the Red UCAV could remain undetected at long distances. Considered herein is the SPY-1D(V) which is part of the AEGIS defense system [41], which is a phased array radar and operates in the S-band (i.e., low frequency for long distance detection). Given the operational nature of the AEGIS system, unclassified information is unavailable. However, open information related to other phased array radar systems which also operate in the S-band can be used, such as the Hughes Air Defense Radar (HADR). The HADR is able to detect a 1 m^2 target at a range of approximately 320 kilometers or 172 nautical miles, which is an idea of what the detection range is going to be. The study assume a fix value for detection range ± 5 nautical miles. This value is 320 kilometers or 172 nautical miles; therefore, the low value is 167 nm and the high value is 177 nm. For a future study, it may be of interest to include the effect of weather conditions.

Number Of Blue UCAVs Launched Per Red

This is a continuous integer factor, and is the number of Blue UCAVs that Blue HVU is going to launch when it detects one incoming Red UCAV. The reason for increasing the number of Blue UCAVs launched is to increase the probability of kill. However, if three Blue UCAVs for each incoming Red are launched and the first launched UCAV kills the Red, the other two are considere lost. Increasing the number of Blue UCAVs per incoming Red UCAV will nominally incur a storage and logistics problem onboard the HVU, but addressing this concern is beyond the scope of this thesis.

3.1.2 Uncontrollable Factors

Uncontrollable factors are those which are considered beyond the engineering design authority or capability of the experimenter, and in simulation are treated as noise factors. These factors

are related to the Red UCAV threat such as their initial numbers and their flight and attack characteristics.

Red UCAV Initial Number

Although the initial number of Red UCAVs is an input for the simulation model, in real life the total number of Red UCAVs would not be known. However, it is known that a UCAV swarm may be used as a first strike to saturate the Blue HVU defenses, e.g., anti-air defenses. Given the adversary's objective, one might assume that at least the same number of AA missiles that a ship can carry must be launched. From open sources, it is known, for example, that the *Arleigh Burke* class destroyer has 90 vertical launcher MK 41s able to shoot 90 AA missiles in the worst case for a Red UCAV swarm attack. As such, the initial number of Red UCAVs is assumed to be any number between 50 and 100. This is a continuous integer factor measured in Red UCAVs.

Red UCAV Dive Angle

As described in Chapter 2 and illustrated in Figure 2.5, the Red UCAV needs to reach a certain angle to change from approach mode to attack mode. This angle is measured in degrees and starts from the vertical of the HVU to the horizon. This is a continuous factor and it can go from 5 to 45 degrees. The impact of this change is that for a small angle (i.e., steeper dive), the Red UCAV stays in the approach mode longer giving the Blue UCAV more time to engage a Red UCAV in that mode. On the other hand, if the angle is greater (i.e., with a shallower angle of attack), the Red UCAV spends less time in approach mode and transition to attack mode sooner increasing the difficulty to engage it.

Red UCAV Speed

From the behavior of Red UCAV, described in Chapter 2, the Red UCAV has three different speeds, one for each mode. No specific publicly available data for such a multi-phase system is readily available. However, it is going to be assumed that the slowest speed is in the Loiter mode, the next faster speed in Approach mode and finally the fastest speed in the Attack mode. These speeds are measured in meters per second and are continuous factors. Although these speeds are inputs for the simulation, ranges for these three speed are define: Loiter speed $30-35$ m/s (or $58-68$ knots), Approach speed $40-45$ m/s (or $77-87$ knots), and Attack speed $50-52$ m/s (or $97-101$ knots).

Red UCAV Endurance

Like the Blue UCAV endurance, this is a continuous factor measured in seconds and is the time from when the Red UCAV is deployed from its base or launching platform until it reaches the

HVU or runs out of battery or fuel. However, it is known that the objective of the Red UCAV swarm is to saturate the HVU defenses so it tries to hit the target before it runs out of battery or fuel. For the simulation's purpose, it is related to the time that the Red UCAV stays in Loiter mode. Available information from the IAI Harpy is used, which dictates an endurance range of $10,000-11,000$ seconds or $166-183$ minutes.

3.1.3 Factor Summary

There are ten factors describe above; however, the Red UCAV Speed factor is divide into three and the Red UCAV Endurance is not consider as a factor because it is assume that each Red UCAV have sufficient endurance to reach its target (i.e., the HVU). The final list of eleven factors is shown in Table 3.1 with the respective ranges and units.

CONTROLLABLE FACTORS				
FACTOR	**TYPE**	**LOW VALUE**	**HIGH VALUE**	**UNITS**
Blue UCAV Speed	Continuous	87	105	Knots
Blast Range	Continuous	10	20	Meters
Blue UCAV Endurance	Continuous	33	66	Minutes
Critical Red	Continuous	1	15	Red UCAV
Detection Range	Continuous	167	177	Nautical Miles
Number of Blue per Red	Continuous	1	3	Blue UCAV
UNCONTROLLABLE FACTORS				
FACTOR	**TYPE**	**LOW VALUE**	**HIGH VALUE**	**UNITS**
Red UCAV Initial Number	Continuous	50	100	Red UCAV
Dive Angle	Continuous	5	45	Degree
Red Loiter Speed	Continuous	58	68	Knots
Red Approach Speed	Continuous	77	87	Knots
Red Attack Speed	Continuous	97	101	Knots
Red UCAV Endurance	Continuous	166	183	Minutes

Table 3.1: Summary Table of Experimental Factors and Levels

In order to create a design of experiments, it is necessary to use normalized or *coded* values for the factors. These coded values are numbers between -1 and 1. To create the coded values the formula below, Equation 3.1, is used [42].

$$X = \frac{Z - \frac{(Z_H + Z_L)}{2}}{\frac{(Z_H - Z_L)}{2}} \tag{3.1}$$

where

$$\begin{aligned} X &= \text{value between -1 and 1} \\ Z_H &= \text{high value of the factor} \\ Z_L &= \text{low value of the factor} \\ Z &= \text{value to code} \end{aligned}$$

For example, to code the Blast Range factor with three levels (high, low and midpoint), with $Z_H = 20$, $Z_L = 10$, and $Z = 10, 20, 15$, the results are:

$$X = \frac{10 - \frac{(20+10)}{2}}{\frac{(20-10)}{2}} = \frac{10 - 15}{5} = \frac{-5}{5} = -1$$

$$X = \frac{20 - \frac{(20+10)}{2}}{\frac{(20-10)}{2}} = \frac{20 - 15}{5} = \frac{5}{5} = 1$$

$$X = \frac{15 - \frac{(20+10)}{2}}{\frac{(20-10)}{2}} = \frac{15 - 15}{5} = \frac{5}{5} = 0$$

So, for the three levels the results are -1, 0 and 1. Now, if the number of levels is increased (X can take any value between -1 and 1), then Equation 3.1 can be used and then solved for Z as shown in Equation 3.2.

$$Z = \frac{-\left[Z_L\left(X - 1\right) - \left(X + 1\right)Z_H\right]}{2} \tag{3.2}$$

With Equations 3.1 and 3.2, the factors described in Figure 3.1 are easily coded when constructing the experimental design matrix. This coded design is translated to actual or *engineering* units for input into the simulation model.

3.2 Design of Experiments

Design of experiments is a methodology to perform experiments varying the input factor in a intelligent way to understand the behavior of the response variable. Montgomery defines an experiment "as a test or series of tests in which purposeful changes are made to the input variables of a process or system so that we may observe and identify the reasons for changes that may be observed in the output response" [42]. He also describes in his text the guidelines for Designing an Experiment:

1. ***Recognition of and statement of the problem.*** This seems to be an obvious step, but most often it is the most important and difficult step. Without a clear understanding of the problem any further steps are useless. For this thesis, the problem statement was discussed in Chapter 2 and the objective of this thesis was described in Chapter 1.

2. ***Selection of the response variable.*** The experimenter should be certain that the selection of the response variable really provides useful information about the process under study. In this thesis, the response variable will be the percentage of killed Red UCAVs by the Blue UCAV defense system. It is a number between 0 and 1.

3. ***Choice of factors, levels, and range.*** The factors considered for the experiments are those that the experimenter may wish to vary in the experiment to see what happens with the response variable. In this thesis, the factors were described at the beginning of this chapter. However, the levels of each factor are defined in the choice of the experimental design.

4. ***Choice of experimental design.*** In selecting the design, the experimenter should keep the objective of the study in mind. For this thesis, the objective is to identify the significant factors that affect the percentage of Red UCAVs killed by the Blue UCAV defense system. First, a screening using a fractional factorial design[6] is carried out to identify the significant factors from the list of eleven factors described above. Then an *I*-optimal design[7] is performed to predict the percentage of kills. As a prediction model, a logistic regression model[8].

[6] A fractional factorial experiment is a variation of the basic factorial design in which only a subset of the runs is used. A factorial design is a combination of all possible combinations of the input factors and their levels.

[7] *I*-optimal design is the one that minimize the value of the average prediction variance.

[8] A Logistic Regression model is used for prediction of probability of occurrence of an event. With the logistic regression, rather than a linear regression, it is possible to ensure to predict a number between 0 and 1.

5. ***Performing the experiment.*** For real life experiments, it is necessary to take into account how to perform the experiments, because the repetition of one experiment may cause problems in gathering unbiased data due to the learning process from each experiment. For this thesis, random seeding of simulation runs ensures independent but repeatable trials, and bias is not relevant since there is no learning process between each experiment.

6. ***Statistical Analysis of the data*** There are many software package designed to assist in data analysis, and most times simple graphical methods are very helpful to understand the results. For the analysis of the data of this thesis, the software JMP® 9 [43] is used extensively.

7. ***Conclusions and recommendations.*** After the data have been analyzed, according to Montgomery, the experimenter must draw practical conclusions about the results and recommend a course of action. In this thesis, all conclusions and recommendations are discussed in Chapter 5.

The output of the experiments, that is, the number of Red UCAVs killed by the Blue UCAVs, requires aggregation of data from the simulation. The simulation software makes it possible to obtain the number easily, but to have it be used as the response variable of interest, which is the *percentage* of Red UCAVs killed, it is necessary to divide the output number by the initial number of Red UCAV for that particular simulation run to get the percentage.

3.2.1 Choice of Design

There are eleven factors in Table 3.1. For the first screening to identify the significant factors, a fractional factorial design with Resolution V[9] is perform. The selection of a Resolution V design is because a second design, i.e., an *I*-optimal design, is select, and it is necessary to make sure that the significant factors selected are indeed the relevant factors and not the aliased ones. For a Resolution V design and eleven factors, it is necessary to perform 128 experiments or runs, and because the simulation is stochastic it is necessary to run one experiment several times to get an average value for that particular experiment or design point. In this thesis, each experiment is perform 50 times, and then the average response is calculated. Therefore, for the first screening experiment, 6400 simulations are perform. In Appendix A the full matrix of the first design with the 128 experiments can be seen.

[9] A Resolution V are designs in which no main effect or two factor interaction is aliased with any other main effect or two factor interaction, but two factor interactions are aliased with three factor interactions

For the second design, an I-optimal design is use with a logistic regression model to analyze the data and to be able to give practical conclusions and recommendations. The second design is explained in greater detail in Chapter 4 along with the analysis of the first screening.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Analysis

4.1 Objective of the Analysis

The objective of this analysis is to identify the significant factors from the list of factors described in Chapter 3. The purpose of identifying the significant factors is to give insight into which are most relevant in the context of the future development of the Blue UCAV defense system. An additional goal is to seek insight into how to use the system in order to achieve a desired level of performance, that is, a probability of kill of the Red UCAV swarm agents.

To achieve these objectives, two designs of experiments (DOEs) are perform. The first one is a screening design using a Resolution V fractional factorial design used to identify the significant factors. For the second DOE, an *I*-optimal design is used to find a prediction model to forecast the probability of kill of the Red UCAV swarm.

4.1.1 Screening Design

To perform the analysis, the statistical software JMP®9 is use. The JMP® software has several features and tools to help to create a DOE and perform the analysis. One of the tools used to perform the analysis is *Fit a Model*. In Figure 4.1, the setup used to perform the analysis of the screening design can be seen. The proposed model to be analyzed is the following:

$$\hat{y} = \beta_0 + \sum_{i=1}^{11} \beta_i X_i + \sum_{i=1}^{11} \sum_{j=1}^{11} \beta_{ij} X_i X_j \tag{4.1}$$

where

$$\begin{aligned} X_i &= \text{factor values described in Chapter 2} \\ \beta_i &= \text{parameter estimates for factor main effects} \\ \beta_{ij} &= \text{parameter estimates for two factor interactions} \end{aligned}$$

Note that quadratic terms are also expressed in Equation 4.1.

JMP®'s *Stepwise* tool is one in which the choice of predictive factors for the proposed model

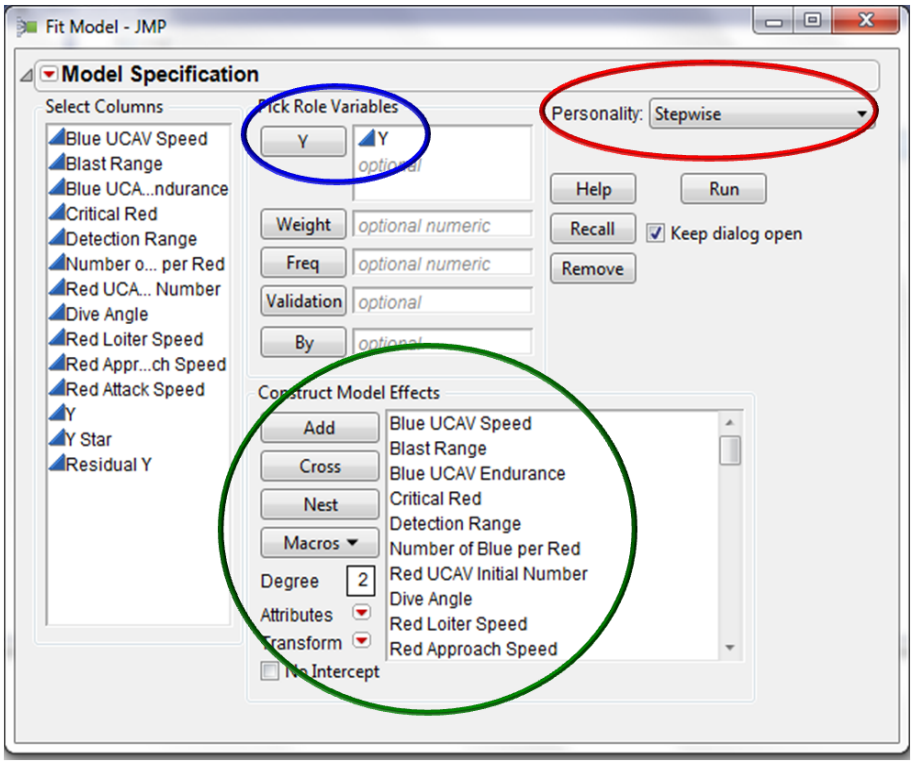

Figure 4.1: Snap shot of the *Fit a Model* screen, where the green oval shows where to put the model effects to be analyzed. In the presented case, a factorial to degree two is used (i.e., main effects and two factor interactions). The red oval is where to choose the type of analysis for the screening. The *Stepwise* tool is used to find the significant factors from the model selected in the green oval. The blue oval shows where to select the response variable, in this case the percentage of Red UCAVs killed.

is carried out by an automatic procedure based on given criteria. These criteria can be selected in the software as shown in Figure 4.2. In this case, the selected stopping rule is *Minimum AICc*, where the Akaike information criterion (AICc) is a measure of the relative value of fit of a statistical model. In other words, AICc describes the tradeoff between bias and variance in model construction. Also, the *Forward* direction is select, which means that the first model to check is the one without variables follow by trying the variables one at the time and retaining the ones that are significant.

It is necessary to mention at this point that 23 more experiments, or design points, are added to the initial screening design because no midpoints had been considered before. Therefore, for the first DOE, 151 experiments are perform with fifty replications for each design point. The complete design matrix with the extra 23 experiments can be found in Appendix A.

After running the *Stepwise* tool, the following factors were determined to be significant factors by the tool:

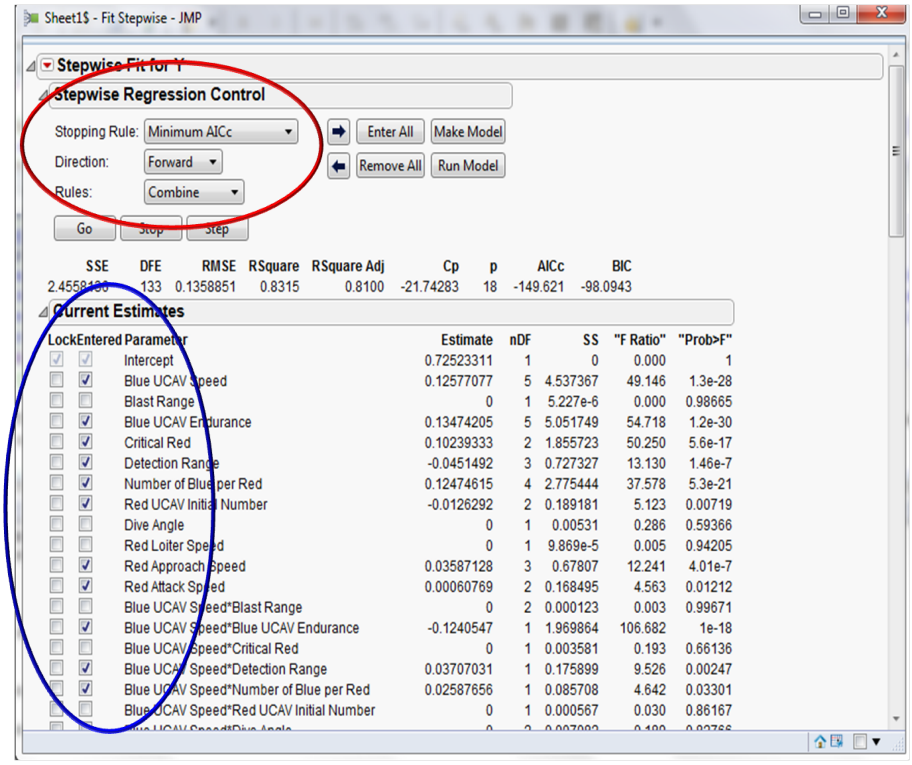

Figure 4.2: Snap shot of the *Stepwise* screen, where the RED oval shows where to specify the criteria to select the significant factors. The BLUE oval is where the significant factors are selected.

- Blue UCAV Speed
- Blue UCAV Endurance
- Critical Red
- Detection Range
- Number of Blue per Red
- Red UCAV Initial Number
- Red Approach Speed
- Red Attack Speed
- Blue UCAV Speed $\times$ Blue UCAV Endurance
- Critical Red $\times$ Critical Red
- Blue UCAV Speed $\times$ Detection Range
- Blue UCAV Endurance $\times$ Detection Range

- Detection Range $\times$ Detection Range
- Blue UCAV Speed $\times$ Number of Blue per Red
- Blue UCAV Endurance $\times$ Number of Blue per Red
- Critical Red $\times$ Number of Blue per Red
- Number of Blue per Red $\times$ Number of Blue per Red
- Blue UCAV Speed $\times$ Red Approach Speed
- Blue UCAV Endurance $\times$ Red Approach Speed
- Critical Red $\times$ Red Approach Speed
- Red UCAV initial Number $\times$ Red Attack Speed

With these factors a standard least square regression model[10] is create to analyze the significant factors.

The model obtained from the least square regression is analyze follow the conditions list below of the model's adequacy, which all pertain to the residuals, i.e., the differences between the predicted value and the observed value:

1. The residuals are normally distributed
2. The residuals have constant variance σ^2
3. The residuals are uncorrelated
4. The residuals have a mean equal to zero

As mentioned before, sometimes plots are good enough to explain or perform analysis. For example, a histogram of the residuals has enough information to check the first and fourth conditions, as is possible to see in Figure 4.3.

To check the constant variance (Condition 2), it is possible to use a plot of *Residual by Predicted* shown in Figure 4.4, where one can see that the residuals have a constant variance over the Y axis, this means that the residuals are equally distributed above and below the mean(mean equal to zero) over the range of the response variable (0 and 1).

[10]The method of least squares is a standard approach to the approximate solution of over determined systems. "Least squares" means that the overall solution minimizes the sum of the squares of the errors made in solving every single equation.

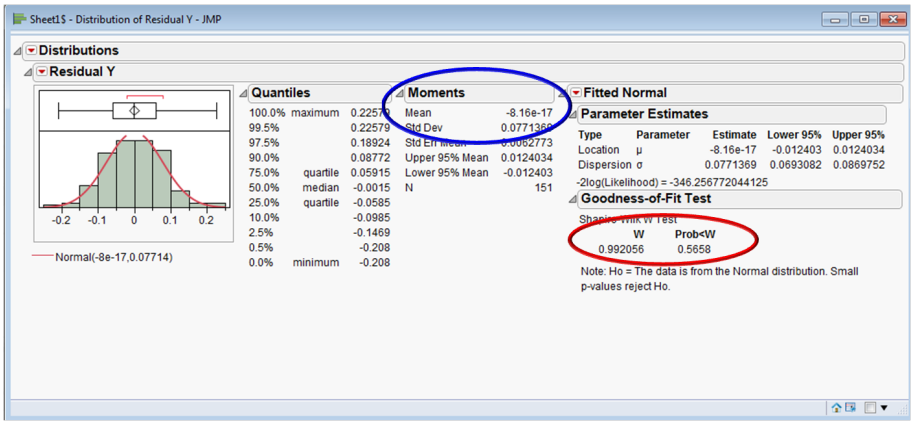

Figure 4.3: Normality analysis, where the RED oval shows that the residuals could be distributed from a normal distribution (Condition 1). The BLUE oval shows that the mean of the residuals is very close to zero (Condition 4).

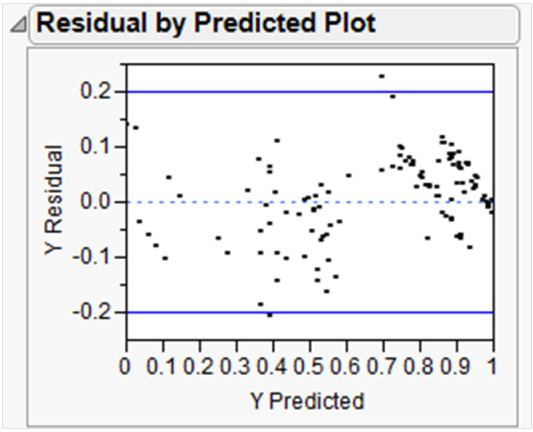

Figure 4.4: Here it is possible to see that the residual of each prediction but two, falls within -0.2 and 0.2 (Blue lines), and also are equally distributed over the range of the response variable.

To check that the residuals are uncorrelated, or there is no evidence of autocorrelation in the residuals, the Durbin-Watson statistic was used. This statistic is used to detect the presence of autocorrelation in the residuals. Frederick *et al.* (2001) [44] explain how to interpret and obtain this statistic using JMP®. When the value of this statistic is near 2.0, this implies that there is no autocorrelation. On the other hand, with a value near 0.0, "there is evidence of positive autocorrelation (high residuals tend to be followed by high residuals, and negative residuals tend to be followed by negative residuals)." Alternatively, a resulting value near 4.0 provides evidence of negative autocorrelation. In Figure 4.5, it is possible to see that the value of the Durbin-Watson statistic is near 2.0; therefore, there is no evidence of autocorrelation, that is, the residuals are uncorrelated.

Each of the four conditions are investigate and shows to be satisfy; therefore, it is possible to

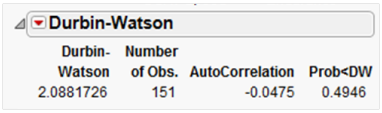

Figure 4.5: The value of the Durbin-Watson statistic is near 2.0; therefore, there is no evidence of autocorrelation, meaning that the residuals are uncorrelated.

say that the model obtain by the least square regression has a good fit to the data. Now it is necessary to check if the model is a good predictor of the data on the summary of the fit table, shown in Figure 4.6. In this figure, it is possible to see the *RSquare* and *RSquare Adj* values, corresponding to the R-squared and adjusted R-squared values of the fit. Recall that *RSquare* provides a measure of how well future outcomes are likely to be predicted by the model. The range of *RSquare* is between zero and one, zero if the model is a bad predictor and one if the model is an excellent predictor; therefore, a higher value is better. The quantity, *RSquare Adj*, penalizes for adding terms that are not helpful to the model, so it is very useful in evaluating and comparing candidate regression models [45]. The range of *RSquare Adj* is also between zero and one, and again a higher value is better than a lower one. However, *RSquare Adj* is always less than or equal to *RSquare*. A good linear prediction model have similar values. In the case presented in this thesis, both of these values are above the 90% mark and their difference approximately 0.01, which is acceptable. Having checked the linear model, the significant factors are shown with their respective estimate values and p-value in Figure 4.7.

4.2 *I*-optimal Design

Having identify the significant factors from the previous design, an *I*-optimal design is perform in order to obtain a good prediction model. Since the response variable is a number between 0 and 1, a variable transformation is perform.The DOE tool within JMP® is use to create the design with this new list of factors (both main effects and interactions), since they exhibited the most significant impact in the responses:

- Blue UCAV Speed
- Blue UCAV Endurance
- Critical Red
- Number of Blue per Red
- Red Approach Speed

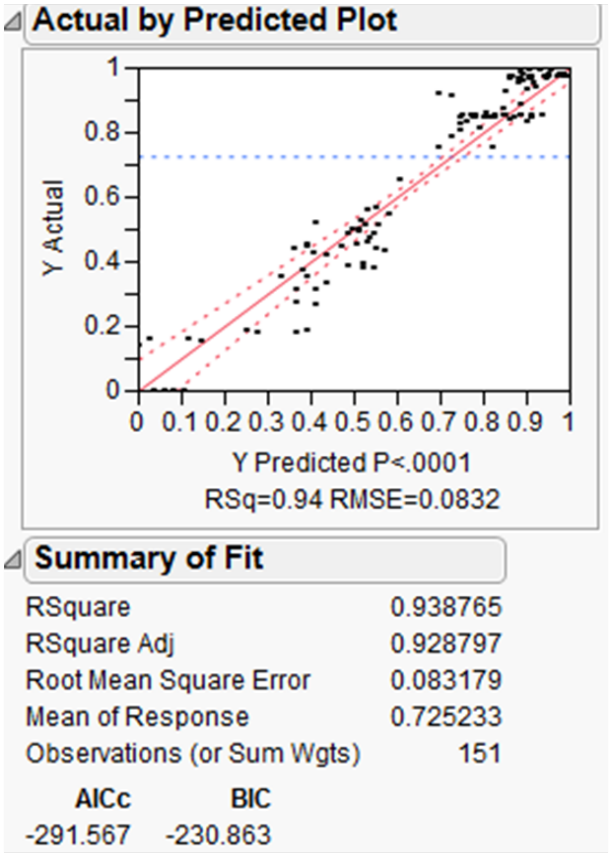

Figure 4.6: This table taken from JMP, shows above a plot of the actual values by the predicted values obtained from the model. If the model were perfect, a line of 45 degrees will be shown instead of the disperse points. The summary table shows the values of *RSquare* and *RSquare Adj*, indicating how well this particular model predicts.

The factor **Detection Range**, described as significant in the previous section, is not consider in this DOE because the launching distance of the Blue UCAV depends on the **Blue UCAV Endurance** and not on the **Detection Range**. Therefore, from the initial list of 11 factors, only the five described in the list above are vary, the rest of the factors stays at their mid values.

This *I*-optimal design has 32 experiments using two (high and low) levels and the same coded values as the screening design. Midpoints were also included in this design. Therefore, the number of experiments totals 43, and again, in each experiment 50 replications are perform. The table of the 43 experiments can be found in Appendix A.

To create an *I*-optimal design in JMP®, it is necessary to propose a model and the list of factors with their ranges. The proposed model is the following:

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	0.984208	0.018763	52.45	<.0001*
Blue UCAV Speed	0.1257708	0.007295	17.24	<.0001*
Blue UCAV Endurance	0.1347421	0.007295	18.47	<.0001*
Critical Red	0.1023933	0.007295	14.04	<.0001*
Detection Range	-0.045149	0.007295	-6.19	<.0001*
Number of Blue per Red	0.1247462	0.007295	17.10	<.0001*
Red UCAV Initial Number	-0.012629	0.007295	-1.73	0.0858
Red Approach Speed	0.0358713	0.007295	4.92	<.0001*
Red Attack Speed	0.0006077	0.007295	0.08	0.9337
Blue UCAV Speed*Blue UCAV Endurance	-0.124055	0.007352	-16.87	<.0001*
Critical Red*Critical Red	-0.137025	0.048499	-2.83	0.0055*
Blue UCAV Speed*Detection Range	0.0370703	0.007352	5.04	<.0001*
Blue UCAV Endurance*Detection Range	0.0473047	0.007352	6.43	<.0001*
Detection Range*Detection Range	-0.050225	0.048499	-1.04	0.3023
Blue UCAV Speed*Number of Blue per Red	0.0258766	0.007352	3.52	0.0006*
Blue UCAV Endurance*Number of Blue per Red	0.0368672	0.007352	5.01	<.0001*
Critical Red*Number of Blue per Red	-0.062045	0.007352	-8.44	<.0001*
Number of Blue per Red*Number of Blue per Red	-0.113559	0.048499	-2.34	0.0207*
Blue UCAV Speed*Red Approach Speed	-0.044152	0.007352	-6.01	<.0001*
Blue UCAV Endurance*Red Approach Speed	-0.04518	0.007352	-6.15	<.0001*
Critical Red*Red Approach Speed	0.0176641	0.007352	2.40	0.0177*
Red UCAV Initial Number*Red Attack Speed	-0.036277	0.007352	-4.93	<.0001*

Figure 4.7: Parameters estimates table screen shot from JMP®. The highlighted (in red) boxes show the significant factors and their coefficient estimate

$$\hat{y} = \beta_0 + \sum_{i=1}^{5} \beta_i X_i + \sum_{i=1}^{5} \sum_{j=1}^{5} \beta_{ij} X_i X_j \tag{4.2}$$

where

$$X_i = \text{factors listed previously}$$
$$\beta_i = \text{parameter estimates for factor main effects}$$
$$\beta_{ij} = \text{parameter estimates for two factor interactions}$$

Note that quadratic terms are also expressed in Equation 4.2.

To analyze the results from these experiments, first a transformation of the response variable is perform using the **LOGIT** function. The **LOGIT** function is used to implement a Logistic Regression Model using the Generalized Logistic Model. With the Logistic Regression it is possible to be sure to predict a number between 0 and 1 and not a negative number or a number above 1, which is the case of the least square regression used in analyzing the first screening

DOE. The **LOGIT** function is the following:

$$y^* = \ln\left(\frac{y}{1-y}\right)$$

where y^* and y are the transformed and original response variables, respectively.

Second, the *Stepwise* tool from JMP® is use to identify significant factors from the proposed model using as a response variable y^*. Then a normal least square regression model is create with the significant factors. In Figure 4.8 it is possible to see the summary of fit for this model.

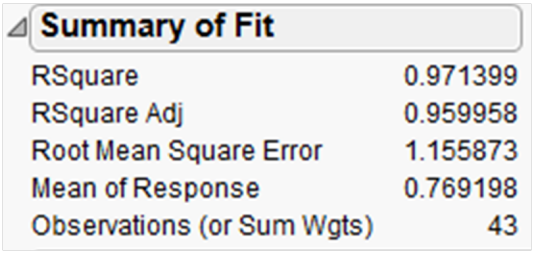

Figure 4.8: Summary of Fit Table for the *I*-optimal design

The results from Figure 4.8 show a good predictor model based on the values of *RSquare* and *RSquare Adj*. However, the model violates the normality assumption of the residuals, as shown in Figure 4.9. The normality assumption is that the residuals can be fitted by a Normal distribution.

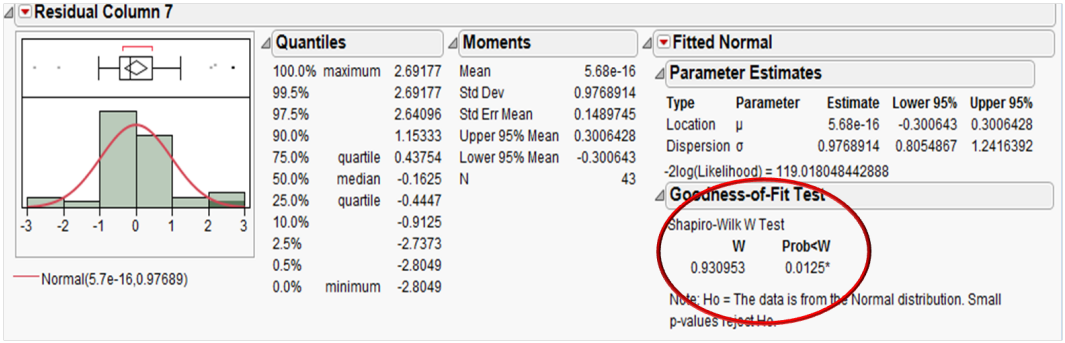

Figure 4.9: Normality assumption fail in *I*-optimal design, the Goodness of Fit test, circled with the RED oval, shows that the null hypotheses (the data comes from a Normal Distribution) is rejected. Therefore, the residuals fail the normality assumption.

Even though the model violates the normality assumption of the residuals, the other assumptions still hold for this model. The mean of the residuals is nearly zero, the residuals are uncorrelated, and the residuals have a constant variance. This model is nevertheless use to perform predic-

tions, knowing that the normality assumption is violate. The reason behind this decision is because it is known that the *I*-optimal design does not cover the entire design space, so maybe the distribution of the residuals is not completely captured by this design, despite the fact that the prediction parameters are acceptable.

To perform the predictions it is necessary to transform the predicted model back to the original model, because now it predicts values between $-\infty$ and ∞ and the original model predicts values between zero and one. To be able to predict values between zero and one it is necessary to have a prediction or fitted formula as used in the following equation, which is simply the inverse of the **LOGIT** function:

$$\hat{y} = \frac{1}{1 + e^{(-\text{prediction formula})}}, \tag{4.3}$$

where the prediction formula from the transformed model is given by the following expression:

$$\begin{aligned} \widehat{y} \;\; = \;\; & 3.982 + 2.389X_1 + 2.586X_2 + 0.870X_3 + 1.468X_4 \\ & +1.998X_5 - 2.472X_1X_2 - 1.902X_1X_5 + 0.410X_2X_4 \\ & -2.182X_2X_5 + 2.048X_3X_4 - 1.768X_2^2 - 2.294X_3^2, \end{aligned} \tag{4.4}$$

where

$$\begin{aligned} X_1 &= \text{Blue UCAV Speed} \\ X_2 &= \text{Blue UCAV Endurance} \\ X_3 &= \text{Critical Red} \\ X_4 &= \text{Number of Blue per Red} \\ X_5 &= \text{Red Approach Speed} \end{aligned}$$

In Figure 4.10 it is possible to see a plot of the predicted values versus the actual values of the experiments used to create the *I*-optimal design.

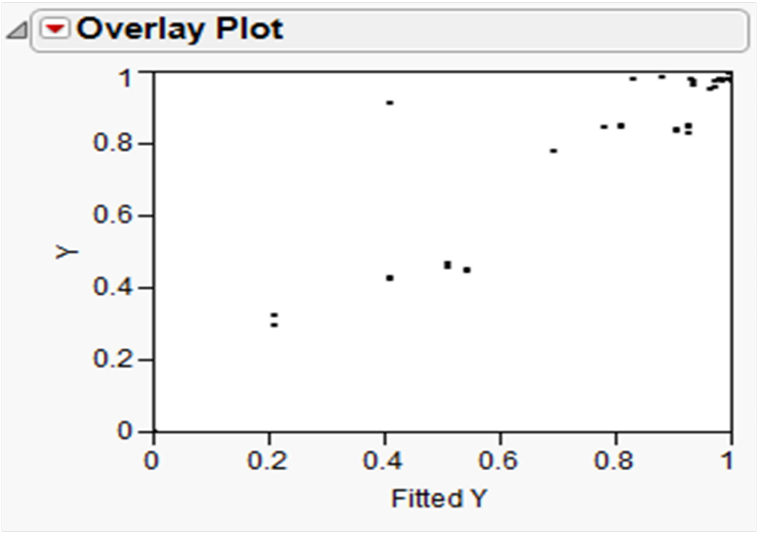

Figure 4.10: Fitted values vs actual values *I*-optimal design, which allows one to check "how well" the prediction model predicts the percentage of killing the Red UCAV swarm. The Y-axis is the values obtained from the simulation and in the X-axis the values obtained from the prediction formula. If the points were aligned in a 45 degrees lines the predicted values will be equal to the values obtained from the simulation.

This prediction model is test against ten experiments created using a space filling technique, the **Latin Hypercube**, in which the five significant factors are use to create ten different experiments with coded values between -1 and 1. These ten experiments are shown in Table 4.1. Then the prediction formula is use to create the prediction and then compare with the actual values obtained from the simulation. The result graph of the predicted values versus the real values is displayed in Figure 4.11. In this figure it is possible to see that there are two predicted values, circled with a red oval, that clearly fail the prediction. The failure of the prediction could be for many reasons, one of them is the small number of experiments carried out to perform the prediction model or the wrong choice of DOE for this data.

Despite this possible weakness of the prediction model and with the goal of obtaining a preliminary intuition, a sensitivity analysis of this prediction model is perform using the *Profiler* plots in JMP®. The *Profiler* plots gives the analyst an idea of how a factor affects the response variable. In this case, five profiler plots are shown in Figure 4.12. Observe that Blue UCAV Speed, Number of Blue per Red, and Red Approach Speed have a linear impact over the response variable, for which, in this untransformed case, the range is between $-\infty$ and ∞. Blue UCAV Endurance and Critical Red factors, on the other hand, have a quadratic impact on the response variable. This fact can also be observed in Equation 4.4, where both factors have a quadratic term in the prediction formula.

Blue UCAV Speed	Blue UCAV Endurance	Critical Red	Number of Blue per Red	Red Approach Speed
0.777778	0.111111	-1	0.333333	0.333333
-0.11111	1	-0.33333	-1	-0.11111
-0.33333	-0.77778	-0.55556	-0.77778	0.777778
0.555556	0.555556	0.777778	-0.11111	1
0.111111	0.777778	0.111111	1	-0.55556
-1	0.333333	-0.11111	0.555556	0.555556
-0.55556	-0.55556	-0.77778	0.111111	-1
0.333333	-1	0.555556	0.777778	0.111111
-0.77778	-0.33333	1	-0.55556	-0.33333
1	-0.11111	0.333333	-0.33333	-0.77778

Table 4.1: Latin Hypercube table

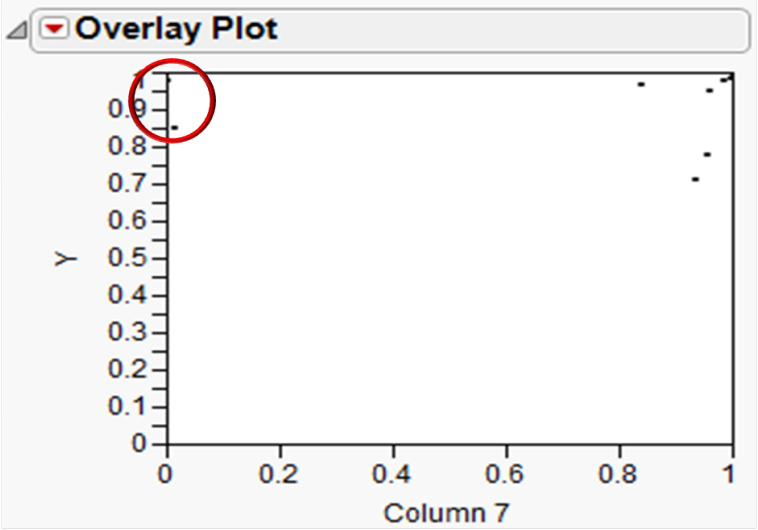

Figure 4.11: Fitted values vs actual values Latin Hypercube test. In this plot it is possible to see how the prediction model predicts experiments obtained from a Latin Hypercube design. The X axis represent the prediction values, and the Y axis represent the real values obtained from the simulation. There are two points that were predicted incorrectly, circled with the RED oval.

In order to clarify and recheck the prediction model, a third experiment is perform. This time a space filling technique was used to create the DOE.

4.3 Space-filling design

The *I*-optimal design is not very satisfactory in order to obtain an accurate prediction model. For this reason a third DOE is perform using the **Sphere Packing** design, which is also available as a

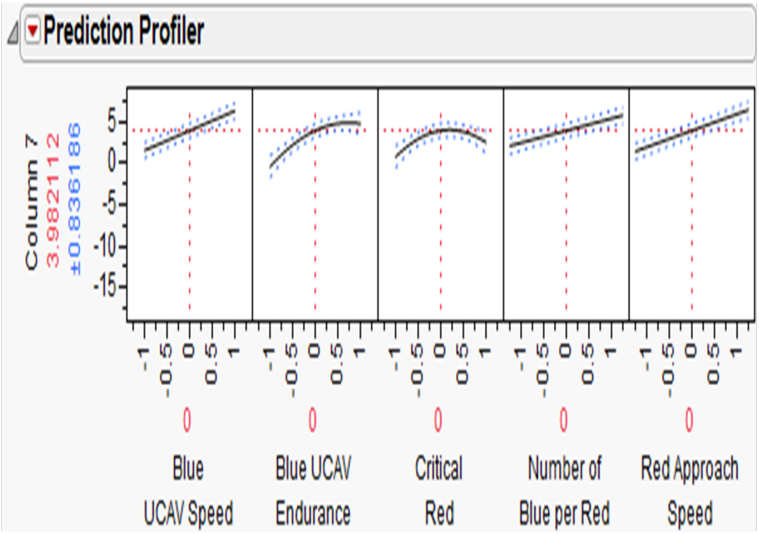

Figure 4.12: Profiler plots from the *I*-optimal design

tool within JMP® to create space filling designs. In this case, 22 experiments are perform using the full eleven initial factors. The list of experiments can be found in Appendix A. After running the experiments, the response variable is transform using the **LOGIT** function. Then using the *Stepwise* function, the significant factors are extract to construct the following prediction model (similar to Equation 4.2):

$$\hat{y} = \beta_0 + \sum_{i=1}^{11} \beta_i X_i + \sum_{i=1}^{11} \sum_{j=1}^{11} \beta_{ij} X_i X_j \tag{4.5}$$

where

$$\begin{aligned} X_i &= \text{factor values described in Chapter 2} \\ \beta_i &= \text{parameter estimates for factor main effects} \\ \beta_{ij} &= \text{parameter estimates for two factor interactions} \end{aligned}$$

Once the simulation experiments is conduct, a standard least square regression model is perform using the transformed response variable from the simulation. This model, unlike the previous models, satisfies all conditions, including the normality assumption of the residuals, constant variance, uncorrelated residuals, and a mean equal to zero. Also, the *RSquare* value is 1 and *RSquare Adj* is 0.999, meaning that the model predicts the data from the experiment with no error. This is clearly a case of an over fitted model, and it seems risky to use it as a prediction

model. As expected, upon checking the prediction formula, the model also unfortunately has the same prediction errors as did the *I*-optimal design. Alternative methods for conducting analysis, such as Gaussian Process models (see [46], [42], and [47]) can be used to analyze this particular set of data. However, the prediction model obtained from the Gaussian Process model is not easy to analyze; for example, using *Excel Solver* to maximize the prediction parameters for the probability of kill response is not straightforward. For this reason, in order to find a good prediction model that allows the use of the *Excel Solver* function (similar to the *Profiler* tool in JMP®), the three described designs are compile into a single design comprising a total of 193 experiments.

The resulted compiled design is a nearly orthogonal design with a maximum correlation of 3.81% as seen in Figure 4.13.

Correlations

	Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detection Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
Blue UCAV Speed	1.0000	0.0035	-0.0059	-0.0107	-0.0129	-0.0238	-0.0077	0.0026	-0.0143	-0.0153	-0.0040
Blast Range	0.0035	1.0000	-0.0207	-0.0040	-0.0067	0.0014	-0.0129	-0.0154	-0.0009	0.0100	0.0068
Blue UCAV Endurance	-0.0059	-0.0207	1.0000	-0.0018	0.0056	-0.0008	0.0124	-0.0149	-0.0060	-0.0023	0.0073
Critical Red	-0.0107	-0.0040	-0.0018	1.0000	0.0252	0.0173	-0.0096	-0.0023	0.0041	0.0054	0.0127
Detection Range	-0.0129	-0.0067	0.0056	0.0252	1.0000	0.0003	0.0051	-0.0012	0.0117	0.0030	-0.0155
Number of Blue per Red	-0.0238	0.0014	-0.0008	0.0173	0.0003	1.0000	-0.0381	0.0126	-0.0012	-0.0018	0.0234
Red UCAV Initial Number	-0.0077	-0.0129	0.0124	-0.0096	0.0051	-0.0381	1.0000	-0.0035	0.0176	-0.0047	0.0047
Dive Angle	0.0026	-0.0154	-0.0149	-0.0023	-0.0012	0.0126	-0.0035	1.0000	-0.0063	-0.0234	0.0034
Red Loiter Speed	-0.0143	-0.0009	-0.0060	0.0041	0.0117	-0.0012	0.0176	-0.0063	1.0000	-0.0004	0.0172
Red Approach Speed	-0.0153	0.0100	-0.0023	0.0054	0.0030	-0.0018	-0.0047	-0.0234	-0.0004	1.0000	0.0062
Red Attack Speed	-0.0040	0.0068	0.0073	0.0127	-0.0155	0.0234	0.0047	0.0034	0.0172	0.0062	1.0000

Figure 4.13: Correlation Matrix of the Compiled design

The same procedure outlined for the previous attempts is perform for this design. First, the response variable is transform using the **LOGIT** function. Then the *Stepwise* tool is use to identify significant factors from the proposed model, incorporating main effects, two-factor interactions, and quadratic terms. The resulting set of significant factors arising in this model is the same as those in the *I*-optimal design, namely *Blue UCAV Speed, Blue UCAV Endurance, Critical Red, Number of Blue per Red*, and *Red Approach Speed*. After that, a standard least square regression model is perform, and a model with *RSquare* of 0.799 and *RSquare Adj* of 0.784 is obtain. However, the residuals obtained from this model fail the normality assumption (again) but satisfy the other conditions. Nevertheless, this model is used to perform the analysis because it shows to result in the best prediction response.

A test of this new prediction model is perform using the same data used to test the *I*-optimal design by effectively checking the difference between the two prediction models. The graphs displayed in Figure 4.14 show on the left the real values from the simulation versus the predicted values from the *I*-optimal design, and on the right graph, the same real values from the

simulation versus the predicted values from the compiled design.

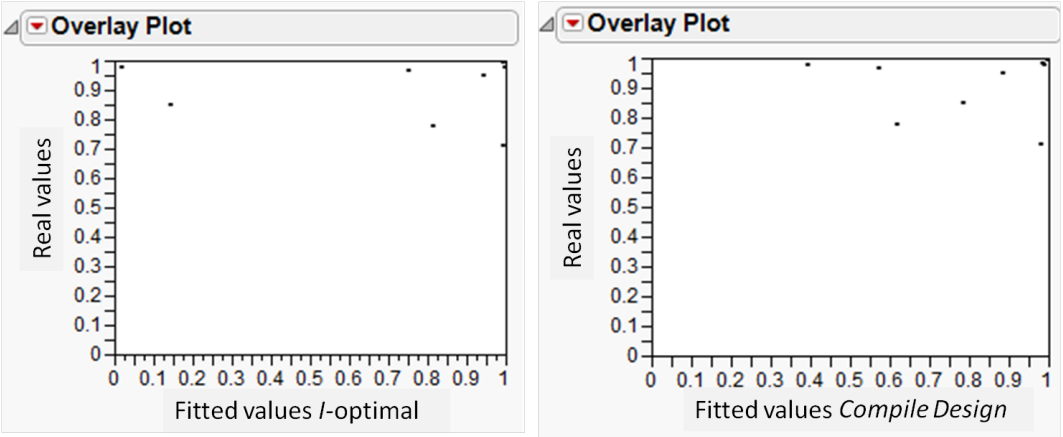

Figure 4.14: The plot on the left shows the predicted values using the prediction model obtained from the *I*-optimal design, where is possible to see there are two predictions that are clearly wrong. The plot to the right shows the same predicted values but using the prediction model obtained from the compiled design. From these plots it is evident that the prediction model obtained from the compiled design is better.

The prediction formula from this model can be described by the following equation:

$$\begin{aligned}\widehat{y} \;=\; & 3.360 + 1.136X_1 + 1.443X_2 + 0.878X_3 + 1.443X_4 \\ & +0.428X_5 - 1.080X_1X_2 + 0.353X_1X_4 - 0.457X_1X_5 \\ & +0.306X_2X_3 + 0.597X_2X_4 - 0.554X_2X_5 + 0.262X_3X_4 - 1.964X_2^2 \end{aligned} \tag{4.6}$$

where

$$\begin{aligned} X_1 &= \text{Blue UCAV Speed} \\ X_2 &= \text{Blue UCAV Endurance} \\ X_3 &= \text{Critical Red} \\ X_4 &= \text{Number of Blue per Red} \\ X_5 &= \text{Red Approach Speed} \end{aligned}$$

Equation 4.6 is analyze using *Excel Solver* to compute the optimized values for the significant factors in order to obtain a 99.9% of probability of kill of the Red UCAV swarm. To verify this prediction model, these optimized values are use as fixed inputs to the simulation model to compare the predicted probability with the real probability obtained from the simulation model. In order to obtain the simulated probability, 10 runs are perform with the optimized values.

Table 4.2 lists these values to obtain a predicted 99.9% probability and compares them with the actual values obtained from the simulation. For completeness, various levels of fidelity are examine. Table 4.3 shows the values to obtain a predicted 90% probability of kill, Table 4.4 shows the value to obtain a predicted 80% probability of kill, and Table 4.5 shows the value to obtain a predicted 50% probability of kill.

Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detection Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
32.4	0.15	57	15	1720	3	75	25	19.44	26.85	30.555

Predicted Probability	Simulated Probability
99.9%	99.06%

Table 4.2: Values to obtain a predicted probability of 99.9%

Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detection Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
29.43	0.15	57	9	1720	2	75	25	19.44	26.52	30.555

Predicted Probability	Simulated Probability
90%	97.86%

Table 4.3: Values to obtain a predicted probability of 90%

Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detection Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
28.93	0.15	59	7	1720	1	75	25	19.44	26.60	30.555

Predicted Probability	Simulated Probability
80%	82.26%

Table 4.4: Values to obtain a predicted probability of 80%

Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detection Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
27.84	0.15	66	3	1720	1	75	25	19.44	26.85	30.555

Predicted Probability	Simulated Probability
50%	56.13%

Table 4.5: Values to obtain a predicted probability of 50%

From Tables 4.2, 4.3, 4.4, and 4.5 one can observe that this model predicts very well the probability generated by the simulation model. Therefore, Equation 4.6 serves as the selected proposed prediction model for further studies.

A sensitivity analysis is conduct for this model using the *Prediction Profilers* from JMP®. In Figure 4.15, one can observe that increasing the values for all significant factors with the exception of *Blue UCAV Endurance*, the response variable will be maximized. It is possible to confirm this by checking the values of the significant factors in Table 4.2, where all but *Blue*

UCAV Endurance are used with its maximum value as input to the simulation model. From Figure 4.15, it is also possible to observe that the factor *Red Approach Speed* is the factor that least affects the response.

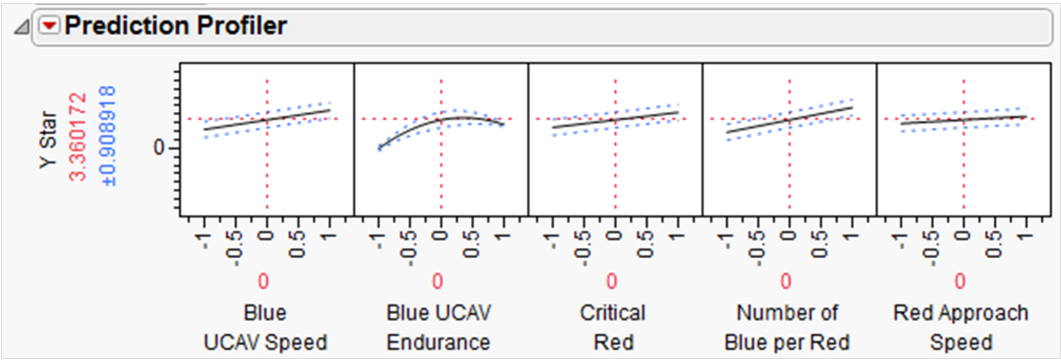

Figure 4.15: Prediction Profiler of the prediction model

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusions and Recommendations

5.1 Conclusions

The main objective of this thesis is to investigate, identify, and analyze the significant factors relevant to the engineering design of a Blue UCAV swarming defense system against an enemy Red UCAV swarm attack. Numerous models and methods for analyses are employed, including statistical design of experiments and applied regression modeling.

Given the future concepts nature of this thesis and the unavailability of open technical specifications for current state-of-the-art systems, this work represents the construction of a reasonable model based on actual data from the open source literature. Though validation is not explicitly performed, model verification studies ensure that the proposed model behave in the intended manner. In future investigations, the agent characteristics and behaviors obtained from open sources can be refined with more accurate or classified information. Incorporation of such modular updates is facilitated by the flexibility of the simulation model built in open source agent-based simulation software called Repast Simphony [3].

Statistical design of experiments is utilized to generate various inputs to the simulation model to elicit relationships between these inputs and the overall response, as measured by the "percentage of Red UCAVs destroyed." From the analysis, the following significant factors for the Blue UCAV defense system are identified:

- Blue UCAV speed
- Blue UCAV endurance
- Critical number of Red UCAVs
- Number of Blue UCAVs launched per Red UCAV

The general relationships are summarized individually below.

Blue UCAV speed

For Blue UCAV speed, the analysis shows that the speed of defensive assets needs to equal or exceed the Red UCAV *approach* speed to remain minimally effective. However, to obtain a higher probability of kill, the Blue UCAV speed is recommended to be comparable, if not greater than, the Red UCAV *attack* speed. Future studies may examine the trade offs (e.g., cost, payload weight, platform agility) required to enable these higher speeds.

Blue UCAV endurance

With respect to the Blue UCAV endurance parameter, high probability results are obtained in the entire range of values between 33 to 66 minutes. However, higher probability results are obtained at the higher level of 66 minutes for endurance. Though intuitive, this relationship may be further explain as follows. If the Blue UCAV is launched when the Red UCAV is still in the approach mode or just changing from loiter to approach mode, given its faster speed (explained above), the Blue UCAV have more time to intercept the Red UCAV and destroy it, whereas alternatively, advance deployments limit the effectiveness due to short battery life spans.

Critical number of Red UCAVs

This factor, Critical Red, that is, the number of direct hits the high value unit can withstand, is very significant in the simulation. Recall that with a Critical Red value of one, the first Red UCAV in the swarm to successfully reach the HVU would terminate the simulation. Yet information pertaining to Blue UCAVs that are previously launched though still engaging the target are not accounted for in the results. In real life, this might be different or might not occur. This is not known because in this thesis, how the Blue UCAV knows the position of the Red UCAV is not discussed. For example, if the information of the position of the Red UCAVs is given by the HVU and one Red UCAV reaches the HVU and damages the communication of the HVU, all the Blue UCAVs that are flying loses their targets. Therefore, Critical Red, should be analyzed in detail, hopefully with real or accurate data before determining the control and communication system of the Blue UCAV.

Number of Blue UCAVs per Red UCAV

The number of Blue UCAVs launched in response to each attacking Red UCAV is also a decisive factor. In the exploratory experiments that are perform at the end of the analysis, all experiments with a probability of kill above 90 percent had the high value of three for this factor, and most responses of probability less than 50 percent had the value of one. This improvement due to redundancy is, however, at the price of transport and deployment. How the Blue UCAVs are to

be stored in the HVU is not discussed in this thesis. It is likely to be infeasible to store 300 Blue UCAVs in case that 100 Red UCAVs were to attack. Prior threat intelligence of how big Red UCAV swarm can be is needed to best determine an appropriate number of Blue UCAVs to be carried by the HVU.

Other factors

The absence of the Blast Range factor in the set of those that are significant is a surprise for the author, given the thought that a larger blast range would possibly provide a tactical advantage. However, after the analysis and further thought, it is understandable why this factor is not significant. First, if the Blue UCAV is not able to reach the Red UCAV (endurance and speed), having a big blast range is useless. Second, given that the Blue speed was greater or equal to the Red approach speed (that is, it almost always reached the Red UCAV), it was a matter of time (endurance) to be within the Blast Range to destroy the Red UCAV.

A number of uncontrollable factors were included in the construction and selection of the design of experiments. Among these factors, only the Red UCAV approach speed was determined to be a significant factor. The resulting conclusion is that the Blue UCAV speed must be at least equal to the Red UCAV approach speed in order for the defensive swarm system to be effective.

5.2 Recommendations

This study is a good starting point to be used as a guideline for agent-based simulation, particularly one with an active and open community. Using Repast Simphony allows the user to simulate and visualize in a three-dimensional environment and also gives more flexibility than other software, such as MANA, to change the agent's behaviors.

Also, this study can be used for shape future designs of the Blue UCAV defense system. However, more accurate information not presently available from open sources must be used. For these following studies, a repository containing the source code of the simulation model has been set up, and for tutorial purposes is also given in Appendix B, as it also provides an illustrative example of how to implement agent-based models in Repast Simphony. These studies should further leverage and implement the *batch* processing capabilities of Repast Simphony when conducting simulation experiments, in order to gather the data in an automated process rather than running experiments individually.

It is also recommended that further validation studies be performed on the proposed simulation model. Such efforts should incorporate the inclusion of more realistic platform motions,

including appropriate flight dynamics for both Blue and Red UCAVs.

5.3 Follow up Studies

The evaluation of different deployment strategies for the Blue UCAV defense system would be a valuable study, as it refers to how and when the Blue UCAV defense system should be launched and operated. For example, a question is whether it is better to launch all available Blue UCAVs upon first detection of the Red UCAV swarm, or to launch according the Red UCAVs' arrival to within a given range? Alternatively, considering there is no endurance constraint, is it better to have a Blue UCAV team loiter over the HVU, waiting for the incoming Red UCAV swarm, or again to launch the Blue UCAVs according the the arrival of Red UCAVs. For this future study, it will be necessary to determine first how the Blue UCAV is going to destroy a Red UCAV, i.e., by impact, explosion, or other method. In contrast, if a loitering Blue UCAV team is infeasible, a possibility is to use another aircraft, such as the manned *Hawkeye* [48] (an early-warning Naval aircraft) or a similar unmanned aerial vehicle, as a "mother ship" to deploy the Blue UCAVs. Though endurance limitations are addressed, such a strategy adds a new constraint: how many Blue UCAVs can be carried by the mother ship? These are some of the topics that should be pursued to continue with this proposed future study.

One important factor, though not considered in this thesis, is the swarming *behavior* of the Blue UCAV defense system. This refers to the ability of the Blue UCAVs to communicate when they are already assigned to an incoming Red UCAV to change their target assignments. For example, if the number of Blue UCAVs per Red UCAV is two, this means that every time that a Red UCAV is within range, two Blue UCAVs will be launched. Consider that one of the Blue UCAVs arrives first and destroys the assigned Red UCAV, leaving the other Blue UCAV idle. For this thesis it was assumed lost, but for a future study, this idle Blue UCAV can be reassigned to a different incoming Red UCAV, and instead of launching two more Blue UCAVs, perhaps only one additional UCAV may be launched. With this new capability of the Blue UCAV defense system, the number of Blue UCAVs that a Blue force should carry can be reduced. To accomplish this new capability, it will be necessary to use a complex control system for the Blue UCAV swarm, increasing its production cost. Therefore, the future analysis should also include the cost factor to determine the trade offs, in terms of cost-effectiveness, between a low-cost UCAV swarm (i.e., large number of simple UCAVs) and a high-cost UCAV swarm (i.e., reduce number of UCAVs but more intelligent).

Another future study is to consider a different type of UAV, capable of carrying larger amounts

of explosive to produce a wider *Lethal Blast Radius* to destroy several incoming Red UCAVs. The purpose of this future study is to first determine if it is possible to produce a large *lethal blast radius* with a single UAV. Secondly, considering that all the attacking Red UCAVs are assumed to be seeking to engage one target, namely the HVU, even though the direction of the threat could be initially omnidirectional, the Red UCAVs must converge to a single position, the HVU. Therefore, it will be interesting to determine, based on the *lethal blast radius*, where this new type of UAV needs to be stationed to kill as many incoming Red UCAV as possible upon detonation. For this future study, further models of the arrival rate for the Red UCAV swarm and better representation of its kinematics (to predicts and estimate target positions) will be relevant. Figure 5.1 graphically depicts the concept for this possible future study.

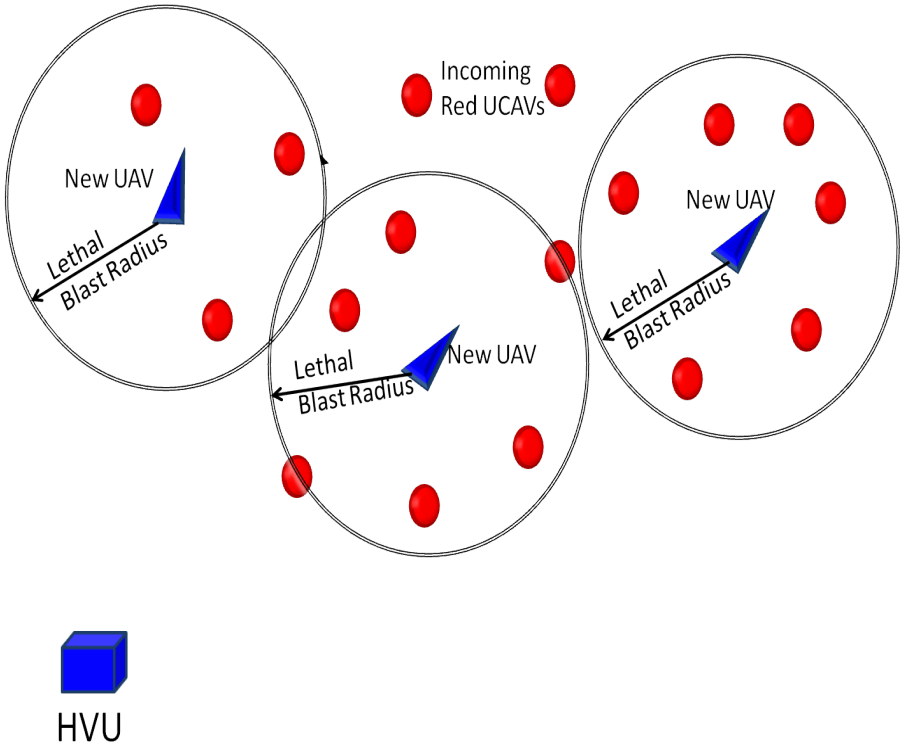

Figure 5.1: Diagram of proposed future study. In this diagram it is possible to see the target of the Red UCAVs, the HVU. All the incoming Red UCAVs must to converge to this target, therefore it is possible to positioned a new type of UAV with a big *Lethal Blast Radius* to destroy the incoming Red.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Israeli-Weapons.com. HARPY. Retrieved June 1, 2011. http://www.israeli-weapons.com/weapons/aircraft/uav/harpy/HARPY.html.

[2] IAI Harpy, 2010. Retrieved June 1, 2011. http://en.wikipedia.org/wiki/IAI_Harpy.

[3] Argonne National Laboratory. Repast Simphony Agent Simulation Toolkit. http://repast.sourceforge.net/index.html.

[4] Robert Andrew Berner. The Effective Use of Multiple Unmanned Aerial Vehicles in Surface Search and Control. Master's thesis in operations research, Naval Postgraduate School, Monterey, CA, December 2004.

[5] Natalie R. Frantz. Swarm Intelligence for Autonomous UAV Control. Master's thesis in electrical engineering, Naval Postgraduate School, Monterey, CA, June 2005.

[6] Kevin M. Schaeffer and Thomas J. Gibbons Jr. Enhancing the Extended Awareness Capabilty of the ESG: Integrating Shotspotter and Cursor on Target Technologies with Unmanned Aerial Vehicles to Enhance the mission Capability of the ESG. Master's thesis in systems technology, Naval Postgraduate School, Monterey, CA, June 2005.

[7] Choon Hon Adrian Teng. Design and Performance Evaluation Study of a Prototype of a Tactical Unmanned Aerial Vehicle. Master's thesis in mechanical engineering, Naval Postgraduate School, Monterey, CA, December 2007.

[8] Michael Sirak. Atk unveils counter uav systems as part of growing portfolio. *Defense Daily*, 2007.

[9] QuinetiQ. QinetiQ and Sula Systems design low cost counter to Tactical UAVs, June 2004. Retrieved June 1, 2011. http://www.qinetiq.com/home/newsroom/news_releases_homepage/2004/2nd_quarter/qinetiq_and_sula_systems.htm.

[10] The Joint Staff. Department of Defense Fiscal Year (FY) 2011 President's Budget. Technical report, Department of Defense, February 2010.

[11] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In Erol Şahin and William M. Spears, editors, *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pp. 10–20. Springer, 2005.

[12] Christian Blum. *Swarm Intelligence: Introduction and Applications*. Springer, Berlin, 2008. ISBN 3540740880.

[13] Robert J. Bamberger Jr, David P. Watson, David H. Scheidt, and Kevin L. Moore. Flight Demonstrations of Unmanned Aerial Vehicle Swarming Concepts. *Johns Hopkins APL Technical Digest*, 27(1):41–55, 2006.

[14] RoboCup.org. RoboCup. Web. Retrieved June 1, 2011. http://www.robocup.org.

[15] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara. RoboCup, A Challenge Problem for AI. *AI Magazine*, 18(1):73–86, 1997.

[16] Hiroaki Kitano, editor. *RoboCup-97: Robot Soccer World Cup I*, volume 1395 of *Lecture Notes in Computer Science*. Springer, 1998. ISBN 978-3-540-64473-6.

[17] William D Shannon. Swarm Tactics and Doctrinal Void: Lessons from the Chechen Wars. Master's thesis in security studies, Naval Postgraduate School, Monterey, CA, June 2008.

[18] John Arquilla and David Ronfeldt. Swarming and the Future of Conflict. Technical report, RAND Corporation, Santa Monica, CA, 2000. http://www.rand.org/pubs/documented_briefings/DB311.

[19] Sze-Tek Terence Ho. Investigating Ground Swarm Robotics Using Agent-Based Simulation. Master's thesis in operations research, Naval Postgraduate School, Investigating Ground Swarm Robotics Using Agent-Based Simulation, December 2006.

[20] Vasileios Lalis. Exploring Naval Tactics with UAVs in an Island complex using Agent-Based Simulation. Master's thesis in operations research, Naval Postgraduate School, Monterey, CA, June 2007.

[21] Travis J Gill. Carrier Air Wing Tactics Incorporating the Navy Unmanned Combat Air System (NUCAS). Master's thesis in operations research, Naval Postgraduate School, Monterey, CA, March 2010.

[22] Chris Anderson. DIYDrones.com. Web, 2011. Retrieved June 1, 2011. http://www.diydrones.com.

[23] Advanced Technologies Incorporated. MALE. Web, 2004. Retrieved June 1, 2011. http://www.advancedtechnologiesinc.com/uav_rotorcraft_male.asp.

[24] MicroPilot. MicroPilot Miniature UAV Autopilots. Web, 2011. Retrieved June 1, 2011. http://www.micropilot.com.

[25] Atalla Buretta, Ryan Fowler, Matthew Germroth, Brian Hayes, Timothy Miller, Evan Neblett, and Matthew Statzer. Low-cost Expendable UAV Project. Final Report, May 2003. http://www.dept.aoe.vt.edu/~mason/Mason_f/CUAVFinalRpt.pdf.

[26] Charles M. Macal and Michael J. North. Tutorial on Agent-Based Modelling and Simulation. *Journal of Simulation*, 4(3):151–162, 2010.

[27] Carmen Zannier. Agent Based Simulation: A Report on Modeling Multiagent Systems as Self-Organized Critical Systems. Technical report, University of Calgary, 2001.

[28] Unmanned aerial vehicle, May 2011. Retrieved June 1, 2011. http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle.

[29] Jerome Bracken, Moshe Kress, and Richard E Rosenthal, editors. *Warfare Modeling*. MORS. John Wiley & Sons, Inc, 1995.

[30] James G Taylor. *Lanchester Models of Warfare*, volume 1. Military Applications Section, Operations Research Society of America, Arlington, VA, 1983.

[31] Sheldon M Ross. *Introduction to Probability Models*. Academic Press, tenth edition, 2009. ISBN 978-0-12-598062-3.

[32] AeroVironment, Inc. Wasp III. Retrieved AeroVironment, Inc. http://www.avinc.com/downloads/Wasp_III.pdf.

[33] Office of the Secretary of Defense. Unmanned Aircraft System Roadmap 2005-2030. Technical report, Department of Defense, 2005.

[34] Dennis McCafferty. UAV Annual Survey. *Special Operations Technology*, 8(6), August 2010.

[35] AAI Corporation. Orbiter Miniature Unmanned Aircraft System, 2009. Retrieved June 1, 2011. http://www.aaicorp.com/pdfs/uas_orbiter07-20-09.pdf.

[36] AAI Corporation. Shadow® 200 Tactical Unmanned Aircraft Systems (TUAS), 2011. Retrieved June 1, 2011. http://www.aaicorp.com/pdfs/shadow_200.pdf.

[37] AAI Corporation. Shadow® 400 Tactical Unmanned Aircraft Systems (TUAS), 2011. Retrieved June 1, 2011. http://www.aaicorp.com/pdfs/shadow400_12-18-09bfinal.pdf.

[38] Dan Johnson. Ruhrstahl/Kramer X-4. Web, 2010. Retrieved June 1, 2011. http://www.luft46.com/missile/x-4.html.

[39] PiLi-5 Short-Range Air-to-Air Missile, October 2008. Retrieved June 1, 2011. http://www.sinodefence.com/airforce/weapon/pl5.asp.

[40] Crotale Low-Altitude Surface-to-Air Missile System. Web, 2008. Retrieved June 1, 2011. http://www.armyrecognition.com/france_french_army_vehicle_missile_systems_uk/crotale_low_altitude_ground_to_air_missile_system_technical_data_sheet_specifications_information_uk.html.

[41] Raytheon Co. Aegis Missile Guidance System SPY-1D(V), 2008. Retrieved June 1, 2011. http://www.raytheon.com/businesses/rtnwcm/groups/public/documents/content/rtn_bus_ids_prod_aegis_pdf.pdf.

[42] Douglas C Montgomery. *Design and Analysis of Experiments*. John Wiley and Sons Inc, 7th edition, 2009. ISBN 0470169907.

[43] SAS Institute Inc. JMP®, Version 9. Cary, NC, 1989-2007. http://www.jmp.com/.

[44] James R Frederick. Multiple Regression in JMP. WEB, March 2001. Retrieved June 1, 2011. http://www.uncp.edu/home/frederick/DSC510/JMPregression.htm.

[45] Douglas C Montgomery. *Introduction to Linear Regression Analysis*. Wiley-Interscience, fourth edition, 2006.

[46] Bradley Jones and Rachel T Johnson. Design and Analysis for the Gaussian Process Model. *Quality and Reliability Engineering International*, 2009.

[47] Carl E Rasmussen. The Gaussian Processes Web Site. WEB, 2011. Retrieved June 1, 2011. http://www.gaussianprocess.org/.

[48] Northrop Grumman. E-2C Hawkeye 2000. Retrieved June 1, 2011. http://www.as.northropgrumman.com/products/e2chawkeye/.

APPENDIX A:
Design Matrices

A.1 Screening Design Tables

It was mentioned in Chapter 2 that the Screening design has 128 experiments plus 23 experiments that include midpoints. To be able to display the entire design matrix, it was separated into five smaller tables. All files relevant to these studies, including electronic versions, can be found at `http://faculty.nps.edu/thchung` under the *Software* section.

A.2 *I*-optimal Design Tables

This design has 43 experiments, and to able to display it was separated in two tables, one with 22 experiments and the other with 21 experiments. The design displayed in these tables are those used as an input in the simulation model. To create the *I*-optimal design, only the significant factors identified in the screening design, i.e., *Blue UCAV Speed, Blue UCAV Endurance, Critical Red*, and *Red UCAV Approach Speed*, were used to build the design. All the other factors were left with their center point values. That is why *Blast Range, Detection Range, Red Initial Number, Dive Angle, Red Loiter Speed and Red Attack Speed* have the same value in all the experiments.

A.3 Sphere Packing Design Table

This design has 22 experiments. The initial 11 factors were used to create this design.

A.4 Latin Hypercube Design Table

This design was created to test the prediction of the other designs. The factors used to create this design were the significant factors identified by the screening design.

Design Point	Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detectio n Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
1	26.85	0.1	33	1	1670	1	50	5	20.98	26.85	31.17
2	26.85	0.1	33	1	1670	1	100	45	17.9	23.76	29.94
3	26.85	0.1	33	1	1670	3	50	45	17.9	23.76	31.17
4	26.85	0.1	33	1	1670	3	100	5	20.98	26.85	29.94
5	26.85	0.1	33	1	1770	1	50	45	17.9	26.85	29.94
6	26.85	0.1	33	1	1770	1	100	5	20.98	23.76	31.17
7	26.85	0.1	33	1	1770	3	50	5	20.98	23.76	29.94
8	26.85	0.1	33	1	1770	3	100	45	17.9	26.85	31.17
9	26.85	0.1	33	15	1670	1	50	45	17.9	26.85	31.17
10	26.85	0.1	33	15	1670	1	100	5	20.98	23.76	29.94
11	26.85	0.1	33	15	1670	3	50	5	20.98	23.76	31.17
12	26.85	0.1	33	15	1670	3	100	45	17.9	26.85	29.94
13	26.85	0.1	33	15	1770	1	50	5	20.98	26.85	29.94
14	26.85	0.1	33	15	1770	1	100	45	17.9	23.76	31.17
15	26.85	0.1	33	15	1770	3	50	45	17.9	23.76	29.94
16	26.85	0.1	33	15	1770	3	100	5	20.98	26.85	31.17
17	26.85	0.1	66	1	1670	1	50	45	20.98	23.76	29.94
18	26.85	0.1	66	1	1670	1	100	5	17.9	26.85	31.17
19	26.85	0.1	66	1	1670	3	50	5	17.9	26.85	29.94
20	26.85	0.1	66	1	1670	3	100	45	20.98	23.76	31.17
21	26.85	0.1	66	1	1770	1	50	5	17.9	23.76	31.17
22	26.85	0.1	66	1	1770	1	100	45	20.98	26.85	29.94
23	26.85	0.1	66	1	1770	3	50	45	20.98	26.85	31.17
24	26.85	0.1	66	1	1770	3	100	5	17.9	23.76	29.94
25	26.85	0.1	66	15	1670	1	50	5	17.9	23.76	29.94
26	26.85	0.1	66	15	1670	1	100	45	20.98	26.85	31.17
27	26.85	0.1	66	15	1670	3	50	45	20.98	26.85	29.94
28	26.85	0.1	66	15	1670	3	100	5	17.9	23.76	31.17
29	26.85	0.1	66	15	1770	1	50	45	20.98	23.76	31.17
30	26.85	0.1	66	15	1770	1	100	5	17.9	26.85	29.94

Table A.1: Screening-1

Design Point	Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detectio n Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
31	26.85	0.1	66	15	1770	3	50	5	17.9	26.85	31.17
32	26.85	0.1	66	15	1770	3	100	45	20.98	23.76	29.94
33	26.85	0.2	33	1	1670	1	50	45	20.98	23.76	31.17
34	26.85	0.2	33	1	1670	1	100	5	17.9	26.85	29.94
35	26.85	0.2	33	1	1670	3	50	5	17.9	26.85	31.17
36	26.85	0.2	33	1	1670	3	100	45	20.98	23.76	29.94
37	26.85	0.2	33	1	1770	1	50	5	17.9	23.76	29.94
38	26.85	0.2	33	1	1770	1	100	45	20.98	26.85	31.17
39	26.85	0.2	33	1	1770	3	50	45	20.98	26.85	29.94
40	26.85	0.2	33	1	1770	3	100	5	17.9	23.76	31.17
41	26.85	0.2	33	15	1670	1	50	5	17.9	23.76	31.17
42	26.85	0.2	33	15	1670	1	100	45	20.98	26.85	29.94
43	26.85	0.2	33	15	1670	3	50	45	20.98	26.85	31.17
44	26.85	0.2	33	15	1670	3	100	5	17.9	23.76	29.94
45	26.85	0.2	33	15	1770	1	50	45	20.98	23.76	29.94
46	26.85	0.2	33	15	1770	1	100	5	17.9	26.85	31.17
47	26.85	0.2	33	15	1770	3	50	5	17.9	26.85	29.94
48	26.85	0.2	33	15	1770	3	100	45	20.98	23.76	31.17
49	26.85	0.2	66	1	1670	1	50	5	20.98	26.85	29.94
50	26.85	0.2	66	1	1670	1	100	45	17.9	23.76	31.17
51	26.85	0.2	66	1	1670	3	50	45	17.9	23.76	29.94
52	26.85	0.2	66	1	1670	3	100	5	20.98	26.85	31.17
53	26.85	0.2	66	1	1770	1	50	45	17.9	26.85	31.17
54	26.85	0.2	66	1	1770	1	100	5	20.98	23.76	29.94
55	26.85	0.2	66	1	1770	3	50	5	20.98	23.76	31.17
56	26.85	0.2	66	1	1770	3	100	45	17.9	26.85	29.94
57	26.85	0.2	66	15	1670	1	50	45	17.9	26.85	29.94
58	26.85	0.2	66	15	1670	1	100	5	20.98	23.76	31.17
59	26.85	0.2	66	15	1670	3	50	5	20.98	23.76	29.94
60	26.85	0.2	66	15	1670	3	100	45	17.9	26.85	31.17

Table A.2: Screening-2

Design Point	Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detection Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
61	26.85	0.2	66	15	1770	1	50	5	20.98	26.85	31.17
62	26.85	0.2	66	15	1770	1	100	45	17.9	23.76	29.94
63	26.85	0.2	66	15	1770	3	50	45	17.9	23.76	31.17
64	26.85	0.2	66	15	1770	3	100	5	20.98	26.85	29.94
65	32.4	0.1	33	1	1670	1	50	45	20.98	26.85	29.94
66	32.4	0.1	33	1	1670	1	100	5	17.9	23.76	31.17
67	32.4	0.1	33	1	1670	3	50	5	17.9	23.76	29.94
68	32.4	0.1	33	1	1670	3	100	45	20.98	26.85	31.17
69	32.4	0.1	33	1	1770	1	50	5	17.9	26.85	31.17
70	32.4	0.1	33	1	1770	1	100	45	20.98	23.76	29.94
71	32.4	0.1	33	1	1770	3	50	45	20.98	23.76	31.17
72	32.4	0.1	33	1	1770	3	100	5	17.9	26.85	29.94
73	32.4	0.1	33	15	1670	1	50	5	17.9	26.85	29.94
74	32.4	0.1	33	15	1670	1	100	45	20.98	23.76	31.17
75	32.4	0.1	33	15	1670	3	50	45	20.98	23.76	29.94
76	32.4	0.1	33	15	1670	3	100	5	17.9	26.85	31.17
77	32.4	0.1	33	15	1770	1	50	45	20.98	26.85	31.17
78	32.4	0.1	33	15	1770	1	100	5	17.9	23.76	29.94
79	32.4	0.1	33	15	1770	3	50	5	17.9	23.76	31.17
80	32.4	0.1	33	15	1770	3	100	45	20.98	26.85	29.94
81	32.4	0.1	66	1	1670	1	50	5	20.98	23.76	31.17
82	32.4	0.1	66	1	1670	1	100	45	17.9	26.85	29.94
83	32.4	0.1	66	1	1670	3	50	45	17.9	26.85	31.17
84	32.4	0.1	66	1	1670	3	100	5	20.98	23.76	29.94
85	32.4	0.1	66	1	1770	1	50	45	17.9	23.76	29.94
86	32.4	0.1	66	1	1770	1	100	5	20.98	26.85	31.17
87	32.4	0.1	66	1	1770	3	50	5	20.98	26.85	29.94
88	32.4	0.1	66	1	1770	3	100	45	17.9	23.76	31.17
89	32.4	0.1	66	15	1670	1	50	45	17.9	23.76	31.17
90	32.4	0.1	66	15	1670	1	100	5	20.98	26.85	29.94

Table A.3: Screening-3

Design Point	Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detectio n Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
91	32.4	0.1	66	15	1670	3	50	5	20.98	26.85	31.17
92	32.4	0.1	66	15	1670	3	100	45	17.9	23.76	29.94
93	32.4	0.1	66	15	1770	1	50	5	20.98	23.76	29.94
94	32.4	0.1	66	15	1770	1	100	45	17.9	26.85	31.17
95	32.4	0.1	66	15	1770	3	50	45	17.9	26.85	29.94
96	32.4	0.1	66	15	1770	3	100	5	20.98	23.76	31.17
97	32.4	0.2	33	1	1670	1	50	5	20.98	23.76	29.94
98	32.4	0.2	33	1	1670	1	100	45	17.9	26.85	31.17
99	32.4	0.2	33	1	1670	3	50	45	17.9	26.85	29.94
100	32.4	0.2	33	1	1670	3	100	5	20.98	23.76	31.17
101	32.4	0.2	33	1	1770	1	50	45	17.9	23.76	31.17
102	32.4	0.2	33	1	1770	1	100	5	20.98	26.85	29.94
103	32.4	0.2	33	1	1770	3	50	5	20.98	26.85	31.17
104	32.4	0.2	33	1	1770	3	100	45	17.9	23.76	29.94
105	32.4	0.2	33	15	1670	1	50	45	17.9	23.76	29.94
106	32.4	0.2	33	15	1670	1	100	5	20.98	26.85	31.17
107	32.4	0.2	33	15	1670	3	50	5	20.98	26.85	29.94
108	32.4	0.2	33	15	1670	3	100	45	17.9	23.76	31.17
109	32.4	0.2	33	15	1770	1	50	5	20.98	23.76	31.17
110	32.4	0.2	33	15	1770	1	100	45	17.9	26.85	29.94
111	32.4	0.2	33	15	1770	3	50	45	17.9	26.85	31.17
112	32.4	0.2	33	15	1770	3	100	5	20.98	23.76	29.94
113	32.4	0.2	66	1	1670	1	50	45	20.98	26.85	31.17
114	32.4	0.2	66	1	1670	1	100	5	17.9	23.76	29.94
115	32.4	0.2	66	1	1670	3	50	5	17.9	23.76	31.17
116	32.4	0.2	66	1	1670	3	100	45	20.98	26.85	29.94
117	32.4	0.2	66	1	1770	1	50	5	17.9	26.85	29.94
118	32.4	0.2	66	1	1770	1	100	45	20.98	23.76	31.17
119	32.4	0.2	66	1	1770	3	50	45	20.98	23.76	29.94
120	32.4	0.2	66	1	1770	3	100	5	17.9	26.85	31.17

Table A.4: Screening-4

Design Point	Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detectio n Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
120	32.4	0.2	66	1	1770	3	100	5	17.9	26.85	31.17
121	32.4	0.2	66	15	1670	1	50	5	17.9	26.85	31.17
122	32.4	0.2	66	15	1670	1	100	45	20.98	23.76	29.94
123	32.4	0.2	66	15	1670	3	50	45	20.98	23.76	31.17
124	32.4	0.2	66	15	1670	3	100	5	17.9	26.85	29.94
125	32.4	0.2	66	15	1770	1	50	45	20.98	26.85	29.94
126	32.4	0.2	66	15	1770	1	100	5	17.9	23.76	31.17
127	32.4	0.2	66	15	1770	3	50	5	17.9	23.76	29.94
128	32.4	0.2	66	15	1770	3	100	45	20.98	26.85	31.17
129	29.6	0.15	50	8	1720	2	75	25	19.45	25.3	30.55
130	32.4	0.15	50	8	1720	2	75	25	19.45	25.3	30.55
131	26.85	0.15	50	8	1720	2	75	25	19.45	25.3	30.55
132	29.6	0.2	50	8	1720	2	75	25	19.45	25.3	30.55
133	29.6	0.1	50	8	1720	2	75	25	19.45	25.3	30.55
134	29.6	0.15	66	8	1720	2	75	25	19.45	25.3	30.55
135	29.6	0.15	33	8	1720	2	75	25	19.45	25.3	30.55
136	29.6	0.15	50	15	1720	2	75	25	19.45	25.3	30.55
137	29.6	0.15	50	1	1720	2	75	25	19.45	25.3	30.55
138	29.6	0.15	50	8	1770	2	75	25	19.45	25.3	30.55
139	29.6	0.15	50	8	1670	2	75	25	19.45	25.3	30.55
140	29.6	0.15	50	8	1720	3	75	25	19.45	25.3	30.55
141	29.6	0.15	50	8	1720	1	75	25	19.45	25.3	30.55
142	29.6	0.15	50	8	1720	2	100	25	19.45	25.3	30.55
143	29.6	0.15	50	8	1720	2	50	25	19.45	25.3	30.55
144	29.6	0.15	50	8	1720	2	75	45	19.45	25.3	30.55
145	29.6	0.15	50	8	1720	2	75	5	19.45	25.3	30.55
146	29.6	0.15	50	8	1720	2	75	25	20.98	25.3	30.55
147	29.6	0.15	50	8	1720	2	75	25	17.9	25.3	30.55
148	29.6	0.15	50	8	1720	2	75	25	19.45	26.85	30.55
149	29.6	0.15	50	8	1720	2	75	25	19.45	23.76	30.55
150	29.6	0.15	50	8	1720	2	75	25	19.45	25.3	31.17
151	29.6	0.15	50	8	1720	2	75	25	19.45	25.3	29.94

Table A.5: Screening-5

Design Point	Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detection Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
1	26.85	0.15	66	15	1720	1	75	25	19.45	26.85	30.55
2	26.85	0.15	66	1	1720	3	75	25	19.45	26.85	30.55
3	26.85	0.15	66	1	1720	1	75	25	19.45	23.76	30.55
4	32.4	0.15	33	1	1720	1	75	25	19.45	23.76	30.55
5	26.85	0.15	33	1	1720	1	75	25	19.45	26.85	30.55
6	26.85	0.15	33	1	1720	3	75	25	19.45	23.76	30.55
7	26.85	0.15	66	1	1720	1	75	25	19.45	23.76	30.55
8	32.4	0.15	33	1	1720	1	75	25	19.45	23.76	30.55
9	32.4	0.15	33	15	1720	3	75	25	19.45	23.76	30.55
10	32.4	0.15	33	1	1720	3	75	25	19.45	26.85	30.55
11	26.85	0.15	33	1	1720	3	75	25	19.45	23.76	30.55
12	26.85	0.15	33	15	1720	1	75	25	19.45	23.76	30.55
13	32.4	0.15	66	1	1720	1	75	25	19.45	26.85	30.55
14	32.4	0.15	66	15	1720	3	75	25	19.45	26.85	30.55
15	32.4	0.15	33	15	1720	1	75	25	19.45	26.85	30.55
16	32.4	0.15	66	1	1720	3	75	25	19.45	23.76	30.55
17	32.4	0.15	33	15	1720	1	75	25	19.45	26.85	30.55
18	26.85	0.15	33	15	1720	1	75	25	19.45	23.76	30.55
19	32.4	0.15	33	15	1720	3	75	25	19.45	23.76	30.55
20	32.4	0.15	66	1	1720	1	75	25	19.45	26.85	30.55
21	26.85	0.15	66	15	1720	3	75	25	19.45	23.76	30.55
22	32.4	0.15	66	1	1720	3	75	25	19.45	23.76	30.55

Table A.6: *I*-optimal-1

Design Point	Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detection Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
23	32.4	0.15	66	15	1720	1	75	25	19.45	23.76	30.55
24	26.85	0.15	66	15	1720	1	75	25	19.45	26.85	30.55
25	26.85	0.15	33	1	1720	1	75	25	19.45	26.85	30.55
26	26.85	0.15	66	15	1720	3	75	25	19.45	23.76	30.55
27	26.85	0.15	33	15	1720	3	75	25	19.45	26.85	30.55
28	32.4	0.15	66	15	1720	3	75	25	19.45	26.85	30.55
29	32.4	0.15	66	15	1720	1	75	25	19.45	23.76	30.55
30	26.85	0.15	66	1	1720	3	75	25	19.45	26.85	30.55
31	26.85	0.15	33	15	1720	3	75	25	19.45	26.85	30.55
32	32.4	0.15	33	1	1720	3	75	25	19.45	26.85	30.55
33	29.6	0.15	50	8	1720	2	75	25	19.45	25.3	30.55
34	32.4	0.15	50	8	1720	2	75	25	19.45	25.3	30.55
35	26.85	0.15	50	8	1720	2	75	25	19.45	25.3	30.55
36	29.6	0.15	66	8	1720	2	75	25	19.45	25.3	30.55
37	29.6	0.15	33	8	1720	2	75	25	19.45	25.3	30.55
38	29.6	0.15	50	15	1720	2	75	25	19.45	25.3	30.55
39	29.6	0.15	50	1	1720	2	75	25	19.45	25.3	30.55
40	29.6	0.15	50	8	1720	3	75	25	19.45	25.3	30.55
41	29.6	0.15	50	8	1720	1	75	25	19.45	25.3	30.55
42	29.6	0.15	50	8	1720	2	75	25	19.45	26.85	30.55
43	29.6	0.15	50	8	1720	2	75	25	19.45	23.76	30.55

Table A.7: *I*-optimal-2

Design Point	Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detection Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
1	26.85	0.20	66	1	1770	1	63	7	20.98	25.98	29.94
2	32.39	0.20	33	15	1670	1	77	5	20.98	24.80	29.94
3	26.85	0.10	66	15	1769	1	80	45	17.90	26.62	30.08
4	27.39	0.10	54	14	1670	2	50	5	17.90	26.85	31.02
5	32.24	0.17	66	1	1670	2	50	5	17.91	23.76	29.94
6	31.99	0.20	34	1	1738	1	50	5	17.90	26.85	31.17
7	32.32	0.10	35	12	1670	1	61	45	17.90	23.76	31.17
8	26.85	0.10	66	2	1670	1	100	5	19.43	23.76	31.17
9	26.85	0.20	34	3	1670	2	50	5	20.97	23.76	31.17
10	28.88	0.20	34	5	1770	1	100	45	20.91	23.76	31.17
11	32.37	0.10	39	1	1770	3	50	45	17.90	26.85	29.94
12	26.86	0.10	33	1	1678	3	100	45	20.98	26.85	31.17
13	28.04	0.20	65	15	1769	3	98	5	17.90	23.76	31.16
14	32.40	0.13	64	14	1770	1	100	5	20.98	26.85	31.17
15	26.85	0.20	33	14	1741	2	50	45	17.90	23.76	29.94
16	32.40	0.10	65	1	1670	1	87	44	20.98	26.37	29.94
17	32.40	0.19	33	15	1725	3	50	45	20.98	26.85	31.16
18	32.40	0.10	33	1	1770	1	100	5	18.33	23.76	30.00
19	26.85	0.10	34	15	1769	3	95	5	20.98	25.51	29.94
20	26.85	0.10	66	9	1769	2	50	45	20.98	23.77	31.15
21	26.85	0.20	33	2	1670	1	99	39	17.90	26.80	29.94
22	32.40	0.20	66	3	1670	3	100	45	17.90	26.58	31.17

Table A.8: *Sphere Packing*

Design Point	Blue UCAV Speed	Blast Range	Blue UCAV Endurance	Critical Red	Detection Range	Number of Blue per Red	Red UCAV Initial Number	Dive Angle	Red Loiter Speed	Red Approach Speed	Red Attack Speed
1	31.78	0.15	51	1	1720	2	75	25	19.45	24.79	30.55
2	29.32	0.15	66	6	1720	1	75	25	19.45	25.48	30.55
3	28.70	0.15	37	4	1720	2	75	25	19.45	24.10	30.55
4	31.17	0.15	59	13	1720	2	75	25	19.45	23.76	30.55
5	29.93	0.15	62	9	1720	3	75	25	19.45	26.16	30.55
6	26.85	0.15	55	7	1720	3	75	25	19.45	24.45	30.55
7	28.08	0.15	40	3	1720	2	75	25	19.45	26.85	30.55
8	30.55	0.15	33	12	1720	3	75	25	19.45	25.13	30.55
9	27.47	0.15	44	15	1720	1	75	25	19.45	25.82	30.55
10	32.40	0.15	48	10	1720	2	75	25	19.45	26.51	30.55

Table A.9: *Latin Hypercube*

APPENDIX B:
Model Source Code

B.1 First Context Creator

This code is used to create the space environment where the agents interact. Here it is possible to modify the size of the simulation space, and also to redefine the loiter area for the Red UCAV agents. Within the code, there are useful comments to help the reader to understand the code.

```
public class firstContextCreator extends DefaultContext<Object>
                implements ContextBuilder<Object> {


   public Context build(Context<Object> context) {
        /**
         * Builds and returns a context. Building a context consists of
         * filling it with agents, adding projects and so forth. When this
         * is called for the master context the system will pass in a
         * created context based on information given in the model.score file.
         * When called for subcontexts, each subcontext that was added
         * when the master context was built will be passed in.
         *
         * @param context
         * @return the built context.
         */
        /* Define the dimensions of the air space */
        int xdim = 3000;          // The x dimension of the physical space
        int ydim = 90;            // The y dimension of the physical space
        int zdim = 3000;          // The z dimension of the physical space

        // Create a new 3D continuous space
        ContinuousSpaceFactoryFinder.createContinuousSpaceFactory(null)
        .createContinuousSpace("airspace3D", context, new RandomCartesianAdder<Object>(),
        new repast.simphony.space.continuous.StickyBorders(), xdim, ydim, zdim);

        ContinuousSpace space = (ContinuousSpace) context.getProjection("airspace3D");


        //  Get the parameters p and then specify the initial numbers of
        //  Red UAV and Blue HVU.
        Parameters p = RunEnvironment.getInstance().getParameters();
        int numRedUAV = (Integer)p.getValue("initialNumberOfRedUAV");
        int numBlueHVU = (Integer)p.getValue("initialNumberOfBlueHVU");

        //Create an array to store the information of each Red UCAV
```

```
        redUAV [] redArray = new redUAV [numRedUAV];
        // Populate the Red UCAV array
        for (int i = 0; i < numRedUAV; i++) {
                redUAV myRedUAV = new redUAV();       // create a new red UCAV
                context.add(myRedUAV);                // add the new red UCAV to airspace
                redArray[i]= myRedUAV;

                // Move the Red UCAVs  to the Loiter area
                space.moveTo(myRedUAV,RandomDraw(2800, 3000),RandomDraw(80, 90),
                RandomDraw(2800, 3000));
        }

        // Set the Red array in the Blue HVU class
        blueHVU.setArray(redArray);

        // Iterate over the number of BlueHVU
        for (int i = 0; i < numBlueHVU; i++) {
                blueHVU myBlueHVU = new blueHVU();     // create a new blue HVU
                context.add(myBlueHVU);                // add the new blue HVU to airspace

                // Move the blue HVU to the initial position
                space.moveTo(myBlueHVU, 100,0,100);

        }

        // If running in batch mode, tell the scheduler when to end each run.
        if (RunEnvironment.getInstance().isBatch()){
                double endAt = (Double)p.getValue("runlength");
                RunEnvironment.getInstance().endAt(endAt);
        }
        return context;
    }


    /**
     * Auxiliar method to get a random position for the Red UAVs
     * @param i lower limit in the axis
     * @param d upper limit in the axis
     * @return position random number within i and d
     */
    private double RandomDraw(int i, int d) {
        double position;
        position = (d-i)*Math.random() + i;
        return position;
    }  // end Auxiliar method

} // end firstContextCreator.java
```

B.2 Blue HVU Class

This code represents the behavior and tasks that the Blue HVU agent performs in the simulation. In the first part the variables are defined. Then in the constructor method, the variables are initialized with the information from the user panel. The other methods are used by Repast in order to perform the simulation.

```
/*
 * Blue HVU Class
 */
public class blueHVU {

        // Define variables
        double px = 0.0;          // position in x
        double py = 0.0;          // position in y
        double pz = 0.0;          // position in z

        double speed;

        double range = 0;         // launching range
        int myCount =0 ;          // keep track of launched blue
        int redHitCount = 0;      // keep track red HITs
        int redKillCount = 0;     // keep track of killed reds

        int bPerR;                // number of blue UCAV per incoming Red

        //Create arrays to store and keep track of the data
        int l;                          // define the length of each array.
        static redUAV [] redArray;      // store the id of each Red UCAV
        double [] distanceArray;        // store the distance to each Red UCAV
        Boolean [] engageArray;         // keep track of the engaged Red UCAVs
        Boolean [] killArray;           // information if the Red UCAV was killed

        // Define agent energy
        double energy;

        /**
         * Constructor of the Blue HVU Agent
         */
        public blueHVU( ){
                this.speed = 0.05;                      // Initialize initial speed
                // Get data from the user panel
                Parameters p = RunEnvironment.getInstance().getParameters();
                this.l = (Integer)p.getValue("initialNumberOfRedUAV");
                int criticalRed = (Integer)p.getValue("criticalRed");
                // Initialize initial energy
                this.energy = 10*criticalRed;           // each Red Hit is 10 units of energy.
                //define the length of each array
                this.distanceArray = new double[l];
                this.engageArray = new Boolean[l];
```

```
        this.killArray = new Boolean [1];
        // Define launching range base on the Blue UCAV speed and Endurance
        double a = (Double)p.getValue("blueUCAVSpeed");
        int b = (Integer)p.getValue("blueEndurance");
        int c = (Integer)p.getValue("launchingRange");
        if(a*b >= c){
                this.range = (double)c;
        }
        else{
                this.range = (a*b)/2;
        } // end if the determine the launching range
        this.range =(Integer)p.getValue("launchingRange");

        // Define number of Blue to launch per incoming Red
        this.bPerR = (Integer)p.getValue("bluePerRed");
} // end blueHVU constructor


/**
 * Populate Red Array
 */
public static void setArray(redUAV[] x){
        redArray = x;
}


// Schedule the step method for agents.  The method is scheduled starting at
// tick one with an interval of 1 tick.  Specifically, the step starts at 1, and
// and recurs at 2,3,4,...etc
@ScheduledMethod(start = 1, interval = 1)
public void step() {
        //Get the context in which the blue HVU resides.
        Context context = ContextUtils.getContext(this);
        ContinuousSpace space = (ContinuousSpace) context.getProjection("airspace3D");
        double heading = RandomDraw(0,2*Math.PI);

        // Get position of the Blue HVU agent and its coordinates on the 3D space to
        // give it to the new blue UAV
        NdPoint blueHVUpoint = space.getLocation(this);
        px = blueHVUpoint.getX();
        py = blueHVUpoint.getY();
        pz = blueHVUpoint.getZ();
        //Populate the information Arrays
        NdPoint redPoint = null;
        for(int m = 0; m < 1; m++){
           if (engageArray[m]!=null){
                // do not update
                }
           else{
              redPoint = space.getLocation(redArray[m]);
              double distBetweenUAV = space.getDistance(blueHVUpoint, redPoint);
              distanceArray[m]= distBetweenUAV;
```

```
                }
            killArray[m] = redArray[m].getBlueFlag();
        }  // end for


        // check distance and create a blue UAV to engage incoming RED UAV
        for (int i = 0; i<l; i++){
            if (killArray[i] == true && redArray[i].getPrint() == true){
                //System.out.println("Red #: " + i +" has been kill");
                redArray[i].setPrint();
                redKillCount++;
            } // end if (killArray[i] == true && redArray[i].getPrint() == true)
            if (distanceArray[i] <= range && engageArray[i] == null){
                engageArray[i]= true;
                //System.out.println("LAUNCHING to RED #: "+ i);
                for (int q = 0; q < bPerR; q++){
                        //RunEnvironment.getInstance().wait(1);
                        blueUAV myblue = new blueUAV((redUAV) redArray[i]);
                        context.add(myblue);
                        space.moveTo(myblue,px,py,pz);
                } // end for (int q = 0; q < bPerR; q++)
                myCount++;
            } // end if (distanceArray[i] <= range && engageArray[i] == null)
        } // end for

        // End Simulation if Number of Red UAV is 0
        if(numRedUAV() == 0){
            System.out.println("Results: Red Killed by Blue:  " + redKillCount +
            "  ; Red HITS: " + redHitCount);
            RunEnvironment.getInstance().endRun();
        }
        //Move the HVU
        move(heading);

        // End Simulation if Number of Red UAV is 0
        if (getEnergy() <= 0 ){
            die();
            System.out.println("Results: Red Killed by Blue:  " + redKillCount +
            "  ; Red HITS: " + redHitCount);
                RunEnvironment.getInstance().endRun();
        } // end if (getEnergy() <= 0 )

    }   // end step() method

    /**
     * Method to move the Blue HVU Agent using as an input the heading
     */
    public void move(double heading) {
        //Get the context in which the RedUAV resides.
        Context context = ContextUtils.getContext(this);
        ContinuousSpace space = (ContinuousSpace) context.getProjection("airspace3D");
```

```
        //Move the agent by vector
        space.moveByVector(this, speed,0,heading,0);

    }  // end move() method


    /**
     *  Public getter for the data gatherer for counting
     */
    public int isBlueHVU() {
        return 1;
    }

    /**
     * Auxiliar method to get a random number between two numbers
     * @param i
     * @param d
     * @return
     */
    public double RandomDraw(int i, double d) {
        double ranNum = (d-i)*Math.random()+i;
        return ranNum;
    }
    /**
     * Getter method for the HVU energy
     */
    public double getEnergy(){
        return energy;
    }
    /**
     * Method to produce a Red UCAV Hit. It will decrease the HVU energy by 10 units
     */
    public void hit(redUAV r){
        int damage = 10;
        int red=0;
        energy = energy-damage;
        for (int m=0;m<1;m++){
                if (redArray[m]==r){
                        red =m;
                }// end if
        }// end for
        redHitCount++;
    }

    /**
     * Method to kill the HVU agent
     */
    private void die() {
        Context context = ContextUtils.getContext(this);
        context.remove(this);
    }
```

```
        /**
         * Method to count the number of flying Red UCAVs
         */
        public int numRedUAV(){
                Context context = ContextUtils.getContext(this);
                ContinuousSpace space = (ContinuousSpace) context.getProjection("airspace3D");
                Iterator list = space.getObjects().iterator();
                Object o = null;
                int nRuav = 0;
                while (list.hasNext()) {
                        o = list.next();
                        if ( o instanceof redUAV) {
                                nRuav +=1;
                        }// end if
                }// end while.has
                return nRuav;
        }

        // Getters to gather data
        public int getMyCount(){return myCount;}
        public int getRedHitCount(){return redHitCount;}
        public int getRedKillCount(){return redKillCount;}

}       // end class redUAV
```

B.3 Blue UCAV Class

This code represents the behavior of the Blue UCAV agent. As in the Blue HVU class definition, first the variables are defined, and then in the constructor method, the variables are initialized. After that, Repast uses the other methods to perform the simulation.

```
@AgentAnnot(displayName = "Agent")
/**
 * Blue UCAV Class
 */
public class blueUAV {
        double px = 0.0;                // position in x
        double py = 0.0;                // position in y
        double pz = 0.0;                // position in z

        double vx = 0.0;                // velocity in x-direction
        double vy = 0.0;                // velocity in y-direction
        double vz = 0.0;                // velocity in z-direction

        double speed = 0.0;             // speed of the Blue UAV
        int endurance;                  // blue UCAV endurance
        int lifeTime = 0;               // keep track of the time the blue UCAV is air born
```

```
double redX = 0.0;                  // X coordinate of the red UAV
double redY = 0.0;                  // Y coordinate of the red UAV
double redZ = 0.0;                  // Z coordinate of the red UAV

redUAV current;                     // Determine the Red UCAV to engage by the Blue UAV

NdPoint redPoint = new NdPoint(redX, redY, redZ);        // Position of the red UAV

double energy = 100.0;  // Define agent energy

double blastRadius;                 // blast radius of the blue UCAV


/**
 * Constructor of the Blue UCAV Agent. As an input uses the  assigned Red UCAV
 * @param rUAV
 */
public blueUAV(redUAV rUAV){
        // Get values from user panel and initialize variables
        Parameters p = RunEnvironment.getInstance().getParameters();
        this.speed = (Double)p.getValue("blueUCAVSpeed");
        this.endurance = (Integer)p.getValue("blueEndurance");
        this.blastRadius = (Double)p.getValue("blastRadius");
        this.current = rUAV;              // set the assigned Red UCAV
}

/**
 * Schedule the step method for agents.  The method is scheduled starting at
 * tick one with an interval of 1 tick.  Specifically, the step starts at 1, and
 * and recurs at 2,3,4,... etc
 */
@ScheduledMethod(start = 1, interval = 1)
public void step() {
        lifeTime++;
        Context context = ContextUtils.getContext(this);
        ContinuousSpace space = (ContinuousSpace) context.getProjection("airspace3D");

        // Get the position of the Blue UCAV agent and its coordinates on the 3D space
        NdPoint blueUAVpoint = space.getLocation(this);
        px = blueUAVpoint.getX();       // The x coordinate on the 3D continuous space
        py = blueUAVpoint.getY();       // The y coordinate on the 3D continuous space
        pz = blueUAVpoint.getZ();       // The z coordinate on the 3D continuous space
        // define a boolean local variable to help to print the data
        boolean killed = false;

        // Check the energy if equal to zero if the Blue UCAV die because explodes
        if(current.getEnergy()==0){
           this.die();
           killed = true;
           return;              // exit this method
        } // end if (current.getEnergy()==0)
```

```
// Check if the Blue UCAV is still flying
if (current.getEnergy() != 0){
   // Get the position of the assigned Red UCAV
   redPoint = space.getLocation(current);
   // Check Real Distance between the UCAVs and set the blue UCAV
   // energy to zero if it misses
   double distToUAV = 100;       // initialize local variable
   if(redPoint==null){  // check if the assigned Red UCAV is dead.
        return;  // exit the method
   }  // end if (redPoint==null)
   // update real distance between UCAVs
   distToUAV = space.getDistance(blueUAVpoint, redPoint);
   // Using the initial position of the assigned Red UCAV add some
   // noise to used as predicted position
   redX = redPoint.getX()+ noise();
   redY = redPoint.getY()+ noise();
   redZ = redPoint.getZ()+ noise();
   // Define a local variable to adjust speed of the Blue UCAV.
   double dist = space.getDistance(blueUAVpoint, redPoint);
   if(dist <= speed){
        speed = dist;  // adjust Blue UCAV Speed
   } // end if(dist <= speed)

   // update the velocity in the 3 coordinates of the blue UCAV
   vx = (px-redX)/dist;
   vz = (pz-redZ)/dist;
   vy = (py-redY)/dist;


   //move the blue UCAV to the predicted position of the Red UCAV
   move();

} // end if (current.getEnergy() != 0)

// create a local variable to help to gather the data
boolean myCatch = false;

// kill the Blue UCAV if its energy is equal to zero
if(current.getEnergy() == 0 && killed == false){
   this.die();
   myCatch = true;
} // end if (current.getEnergy() == 0 && killed == false)

// Check distance to kill the assigned Red UCAV
if (current.getEnergy() != 0  && killed == false){
   double rangeToTarget = space.getDistance(space.getLocation(this),
                                space.getLocation(current) );
   // if the distance between UCAVs is less than the Blast Radius.
   if (rangeToTarget <= blastRadius ){
        this.die();              // Blue UCAV explodes
        myCatch = true;
        // Probability that the Red UCAV die with the explosion
```

```
                if (Math.random() <= .85){
                    current.die();                          // kill the Red UCAV
                    current.setBlueFlag();                  // Inform to the HVU
                } // end if (Math.random() <= .85)
            }  // end if(rangeToTarget <= blastRadius )
        }  // end if (current.getEnergy() != 0  && killed == false)

        // kill the agent if its energy is equal to zero or lifetime
        if(myCatch == false &&(energy == 0 || lifeTime >= endurance)){
            this.die();                                     // kill the Blue UCAV
        } // end if(myCatch == false &&(energy == 0 || lifeTime >= endurance))

}// end step method



/**
 * This method is used to move the Blue UCAV agent
 */
public void move() {
     //Get the context in which the blueUAV resides.
     Context context = ContextUtils.getContext(this);
     ContinuousSpace space = (ContinuousSpace) context.getProjection("airspace3D");
     // Move by vector
     space.moveByDisplacement(this,-vx*speed,-vy*speed,-vz*speed);

} // end move() method

/**
 * Method to kill an Agent
 */
private void die() {
     Context context = ContextUtils.getContext(this);
     context.remove(this);
} // end die

/**
 * Method to decrease the energy of the Agent
 */
private double setEnergy(double i) {
     return energy = i;
} // end setEnergy

/**
 * Method to create the noise to the Red UCAV position
 * @return a number between 0 and 1/10000.
 */
private double noise(){
     double noise = Math.random()/10000;
     return noise;
} // end noise
```

```
        /**
         * Public getter for the data gatherer for counting
         * @return
         */
        public int isBlueUAV() {
             return 1;
        } // end isBlueUAV

}// end Blue UCAV class
```

B.4 Red UCAV Class

This is the code that defines the model of the Red UCAV agent behavior. Again, similar to the Blue HVU and Blue UCAV classes, first the variables are defined and then in the constructor method the variables are initialized. Repast uses the other methods described in this code to perform the simulation.

```
@AgentAnnot(displayName = "Agent")


/**
 *
 * Red UCAV class
 *
 */
public class redUAV {

        double px = 0.0;                // position in x
        double py = 0.0;                // position in y
        double pz = 0.0;                // position in z
        double vx = 0.0;                // velocity in x-direction
        double vy = 0.0;                // velocity in y-direction
        double vz = 0.0;                // velocity in z-direction
        double loiterSpeed;             // loiter speed of the red UCAV
        double approachSpeed;           // approach speed of the red UCAV
        double attackSpeed;             // attack speed of the red UCAV
        double currentSpeed;            // speed to keep be used on the simulation
        double energy = 100.0;          // Define agent energy
        double time;                    // time count to engage [s]
        double aAngle;                  // Attack angle of the Red UAV Swarm
        double engage;                  // Engagement time
        blueHVU hvu;                    // get the value of the agent HVU
        Boolean blueFlag = false;       // boolean to keep track of the data in the HVU
        Boolean print = true;           // boolean to help with gathering data
```

```
 /**
  * Constructor of the Red UCAV agent
  */
public redUAV(){
        // Get values from user panel and initialize variables
        Parameters p = RunEnvironment.getInstance().getParameters();
        this.aAngle = (Double)p.getValue("diveAngle");
        this.loiterSpeed = (Double)p.getValue("loiterSpeed");
        this.approachSpeed = (Double)p.getValue("approachSpeed");
        this.attackSpeed = (Double)p.getValue("attackSpeed");
        this.currentSpeed = loiterSpeed;            // set current speed to loiter speed
        this.engage = RandomDraw();                 // initialize the engagement time
}               // end redUAV constructor

//Schedule the step method for agents.  The method is scheduled starting at
// tick one with an interval of 1 tick.  Specifically, the step starts at 1, and
// and recurs at 2,3,4,... etc
@ScheduledMethod(start = 1, interval = 1)
   public void step() {
        //Get the context in which the RedUAV resides.
        Context context = ContextUtils.getContext(this);
        ContinuousSpace space = (ContinuousSpace) context.getProjection("airspace3D");

        // Set random movement for the red UAV in Loiter Mode
        vx = RandomDraw(-1,1);
        vy = 0;
        vz = RandomDraw(-1,1);
        // Initialize distance variables
        double distToHVU=2;
        double distToUAV=2;
        // get the simulation time
        time = RunEnvironment.getInstance().getCurrentSchedule().getTickCount();
        // Check if engage or not (change from Loiter to Approach mode)
        // if the simulation time is greater to the engage time of each agent then ENGAGE
        if (time > engage){
                // Increase speed to Approach mode speed
                currentSpeed = approachSpeed;
                // Get the Red UCAV Coordinates from the space
                NdPoint point = space.getLocation(this);
                px = point.getX();      // The x coordinate on the 3D continuous space
                py = point.getY();      // The y coordinate on the 3D continuous space
                pz = point.getZ();              // The z coordinate on the 3D continuous space

                //Find the position of the Blue HVU
                Iterator list = space.getObjects().iterator();
                Object o = null;
                NdPoint bluePoint = null;
                while (list.hasNext()) {
                        o = list.next();
                        if ( o instanceof blueHVU ) {
                                hvu = (blueHVU)o;
                                bluePoint = space.getLocation( o );
```

```
                }// end if for BlueHVU
            } // end while

            // check if the HVU has been destroy
            if(hvu.getEnergy()==0){
                return;     // exit the method
            } //end if HVU is destroy

            //Calculate the Red UCAV velocity vector
            distToHVU = space.getDistance(point, bluePoint);
            vx = (px-bluePoint.getX())/distToHVU;
            vz = (pz-bluePoint.getZ())/distToHVU;
            vy = 0;         // in approach mode, the Red UCAV have a fixed altitude

            // if the angle is equal or less than aAngle dive into blue HVU
            // (change from approach to attack mode)
            if(Math.cos(Math.toRadians(aAngle)) <= (py-bluePoint.getY())/distToHVU){
                vy = (py-bluePoint.getY())/distToHVU;
                //Increase speed to attack mode speed
                currentSpeed = attackSpeed;
            } // end if dive into blueHVU

    }// end if engage

    //adjust the speed to hit the HVU
    if (distToHVU <= currentSpeed){
        currentSpeed = distToHVU;
    }
    //Move the RedUAV
    move();

    //check distance to HVU
    if(distToHVU <= 0.01){
        setEnergy(0);           // Red UCAV reach the HVU
        hvu.hit(this);          // hit the HVU
    } //end if (distToHVU <= 0.01)

    // kill the agent
    if(energy<=0){
        die();
    }  // end if(energy<=0)


}  // end step() method

/**
 * Method to move the Red UCAV agent
 */
public void move() {
    //Get the context in which the RedUAV resides.
    Context context = ContextUtils.getContext(this);
    ContinuousSpace space = (ContinuousSpace) context.getProjection("airspace3D");
```

```
        // Move by Displacement
        space.moveByDisplacement(this,-vx*currentSpeed,-vy*currentSpeed,-vz*currentSpeed);
    }           // end move() method


    /**
     * Public getter for the data gatherer for counting
     * @return
     */
    public int isRedUAV() {
        return 1;
    }  // end isRedUAV

    /**
     * Auxiliar method to get a random number between two numbers
     * @param i
     * @param d
     * @return
     */
    private double RandomDraw(int i, double d) {
        double ranNum = (d-i)*Math.random()+i;
        return ranNum;
    }  // end RandomDraw

    /**
     * Auxiliar method to get a random number according certain distribution
     * @return
     */
    private double RandomDraw() {
        double engage = Math.random()*100 +15;
        return engage;
    }  // end RandomDraw

    /**
     * Setter method for Energy
     */
    public double setEnergy(double i) {
        return energy = i;
    }
    /**
     * Getter method for Energy
     */
    public double getEnergy(){
        return energy;
    }
    /**
     * Method to kill the Red UCAV Agent
     */
    public void die() {
        Context context = ContextUtils.getContext(this);
        context.remove(this);
```

```
    }
    /**
     * Getter method for Red UCAV Speed
     */
    public double getSpeed(){
        return currentSpeed;
    }
    /**
     * Getter method for Red UCAV x vector velocity
     */
    public double getXvel(){
        return vx;
    }
    /**
     * Getter method for Red UCAV y vector velocity
     */
    public double getYvel(){
        return vy;
    }
    /**
     * Getter method for Red UCAV z vector velocity
     */
    public double getZvel(){
        return vz;
    }
    /**
     * Setter method for Blue flag to gather data in the HVU agent
     */
    public Boolean setBlueFlag(){
        blueFlag = true;
        return blueFlag;
    }
    /**
     * Getter method for Blue flag to gather data in the HVU agent
     */
    public Boolean getBlueFlag(){
        return blueFlag;
    }
    /**
     * Getter method to gather data in the HVU agent
     */
    public Boolean getPrint(){
        return print;
    }
    /**
     * Setter method to gather data in the HVU agent
     */
    public void setPrint(){
        print = false;
    }
}  // end class redUAV
```

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia
2. Dudly Knox Library
   Naval Postgraduate School
   Monterey, California
3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California
4. Directory, Training and Education, MCCDC, Code C46
   Quantico, Virginia
5. Marine Corps Tactical System Support Activity (Attn: Operations Officer)
   Camp Pendleton, California
6. Director, Studies and Analysis Division, MCCDC, Code C45
   Quantico, Virginia
7. Dr. Timothy H. Chung
   Naval Postgraduate School
   Monterey, California
8. Dr. Michael Atkinson
   Naval Postgraduate School
   Monterey, California
9. CPT. Francisco J. Hederra
   Direccion de Investigacion, Programas y Desarrollo de la Armada
   Armada de Chile
   CHILE
10. CAPT Jeffrey Kline, USN(ret.)
    Naval Postgraduate School
    Monterey, California